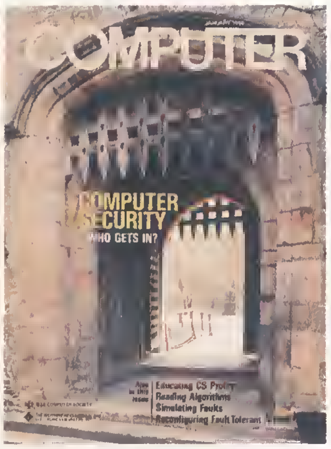
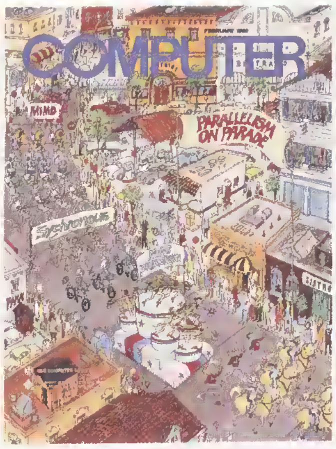
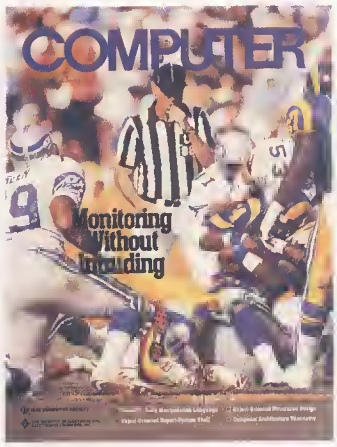
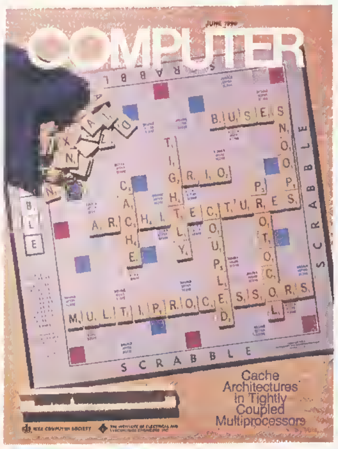
DECEMBER 1990

# COMPUTER









COMPUTER
SEPTEMBER 1990
Formal Methods
Prelude to Virtuoso Software
Companion article in September IEEE Software and IEEE Transactions on Software Engineering

COMPUTER
Dependability
Fault Tolerant Concurrent Software
Fault Tolerant Clock Synchronization
Built-in Self-Test RAM

## In this issue:

- Scientific Programming
- Compiling Scientific Code
- Parallel Computation
- Engineering Philosophies
- Common Benchmarks

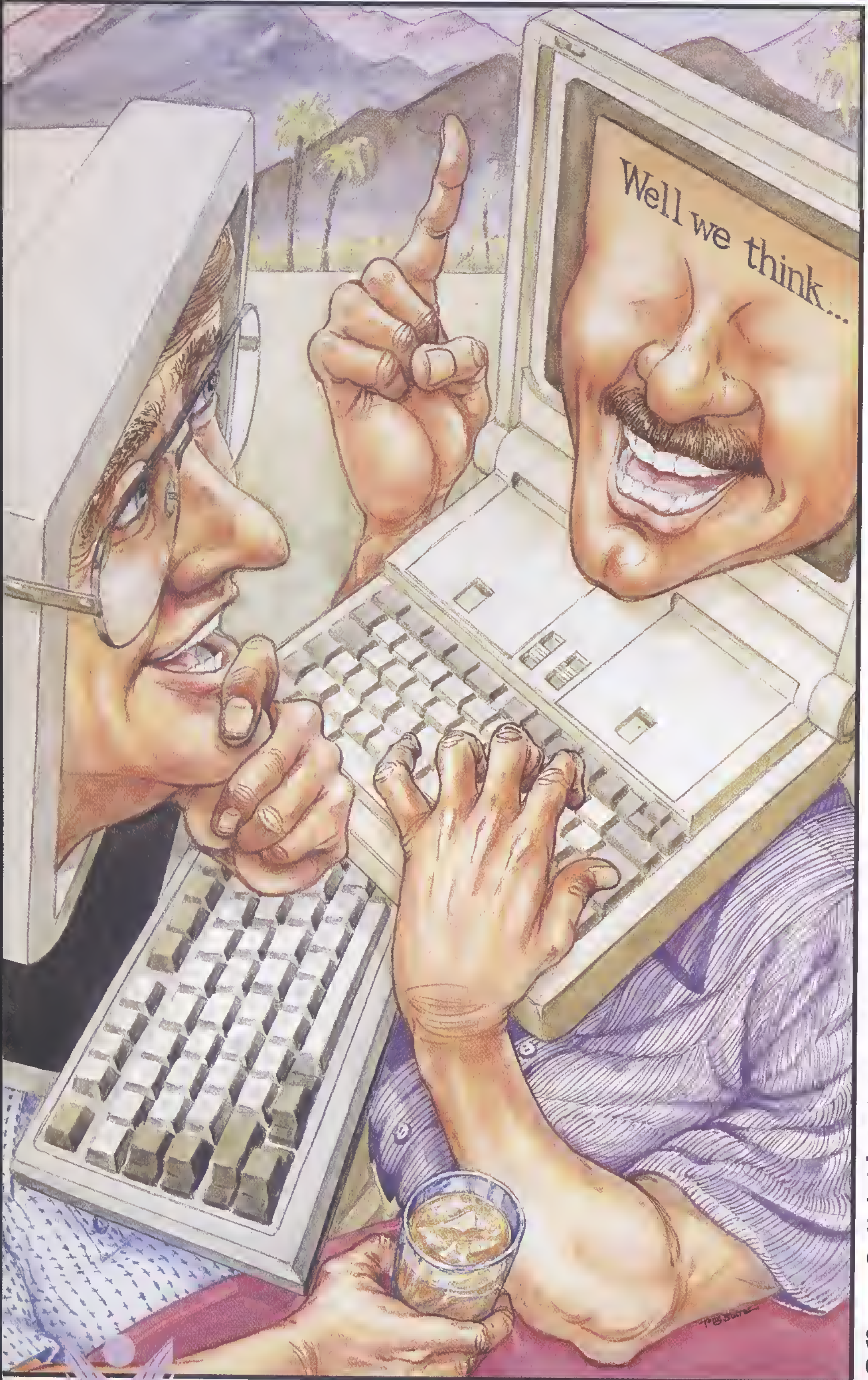
## 1990 Annual Index

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

T ♦ E ♦ N ♦ T ♦ H   A ♦ N ♦ N ♦ I ♦ V ♦ E ♦ R ♦ S ♦ A ♦ R ♦ Y

# SEE WHAT'S ON OUR MINDS



Conferences sure have changed. Lately it seems, the more specialized the better. But most of us are having to learn and apply a broad range of technologies on a daily basis. What's important to us too, is the chance to exchange ideas with our peers, to see what's happening on the other side of the fence, and be exposed not only to new ideas, but different ways of thinking about the old.

And that's where we come in. Over the past ten years we've developed a conference that tries very hard to cover a broad range of topics and deals with the interesting problems of computers, communications, AND computers communicating. And we've tried to keep that balance between new theories and practical application.

We've learned (from talking to people like you) that sitting all day listening to paper presentations isn't all that a conference should be. So we've added state-of-the-art exhibits showcasing the relationships between computers and communications. And lots more panel sessions where you have a chance to debate the latest hot topics with experts in the field. A wide array of full-day tutorials gives you a chance for some intense training on new techniques.

We even have our own "communication" room, where each attendee has access to computers and networks world wide.

We hold this conference in Arizona, and without bragging, at what happens to be the best time of year to visit the desert.

So come to Scottsdale, and help us celebrate our tenth anniversary by making this our best conference to date. We can't do it without you.

Contact us for a poster, a brochure, or more detailed information. Because, even though we're computer heads, how are we going to communicate if we don't get together and talk?

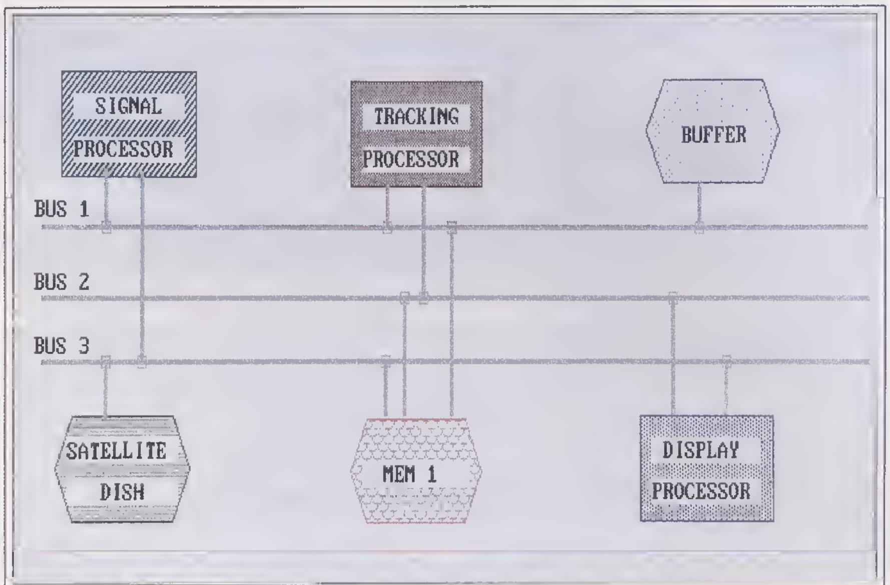
**March 27-30, 1991**
**Scottsdale, Arizona**

**Contact: Mary Murphy-Hoye**
(602) 554-5257 or mhoye@fa.intel.com

**Sponsored by the IEEE Communication Society**

INTERNATIONAL PHOENIX CONFERENCE ON COMPUTERS & COMMUNICATIONS

**NEW FOR NETWORK ANALYSTS AND SYSTEMS ENGINEERS**



Realistic simulation of your network or embedded computer system--quick results, no programming

# NETWORK II.5 now predicts performance of computer-communication systems

## Free trial and, if you act now, free training

NETWORK II.5 uses simulation to predict your network performance. You simply describe your network and workload.

**Animated simulation follows immediately--no programming delays.**

**Easy-to-understand results**

You get an animated picture of your network. System bottlenecks and changing levels of utilization are apparent.

Seeing your network animated increases everyone's understanding of its operation and builds confidence in your results.

Your reports show response times, messages delivered, messages lost, device utilization, and queueing statistics.

**Computers with NETWORK II.5**

NETWORK II.5 is available for most PC's, Workstations, and Mainframes.

**Your network simulated**

You can analyze embedded or distributed computer systems, or other computer-communication networks. Industry standard protocols such as FDDI and IEEE Standard 802.X are built-in. Others can be modeled.

You can easily study the effect of changing network parameters or even network protocols.

You can simulate some portions of the network at a detailed level and others at a coarser level.

**Free trial information**

The free trial contains everything you need to try NETWORK II.5® on your computer. For a limited time we also include free training **--no cost, no obligation.**

Call Paul Gorman at (619) 457-9681, Fax (619) 457-1184. In Europe, call Nigel McNamara, in the UK, on (081) 332-0122, Fax (081) 332-0112.

## Free trial offer

See for yourself how NETWORK II.5 quickly answers network performance questions.

**Limited offer--Act now for free training.**

☐Send details on your University Offer.

Name

Organization

Address

City State Zip

Telephone Fax

Computer Op. Syst.

**Return to:** IEEE COMP
CACI Products Company
3344 North Torrey Pines Court
La Jolla, California 92037
**Call Paul Gorman at (619) 457-9681.**
**Fax (619) 457-1184.**

**In Europe:**
CACI Products Division
Palm Ct., 4 Heron Square
Richmond-Upon-Thames
Surrey TW9 1EW, UK
**Call Nigel McNamara on (081) 332-0122.**
**Fax (081) 332-0112.**

# COMPUTER

December 1990 Published by the IEEE Computer Society Vol. 23, No. 12

## ARTICLES

**Circulation:** *Computer* (ISSN 0018-9162) is published monthly by the IEEE Computer Society, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264; phone (714) 821-8380. IEEE Computer Society Headquarters, 1730 Massachusetts Ave. NW, Washington, DC 20036-1903; IEEE Headquarters, 345 East 47th St., New York, NY 10017. Annual subscription included in society member dues. Nonmember subscription rates: available upon request. Single copy prices: members $10.00; nonmembers $20.00. This magazine is also available in microfiche form.

**Postmaster:** Send undelivered copies and address changes to *Computer*, IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08855. Second class postage is paid at New York, New York, and at additional mailing offices.

BPA

## DEPARTMENTS

Cover design: Jay Simpson, Design & Direction

## In the next issue
Experimental research in computer architecture

**COMPUTER**

10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264

**Editor-in-Chief**
Bruce D. Shriver, University of Southwestern Louisiana

**Associate Technical Editor**
Jon T. Butler, Naval Postgraduate School

**Editorial Board**
Dharma P. Agrawal, North Carolina State University
Dennis R. Allison, Stanford University
Fletcher Buckley, GE Aerospace
Scott Davidson, AT&T
Richard Eckhouse, MOCO
Michael Evangelist, Andersen Consulting
Scott E. Fahlman, Carnegie Mellon University
John R. Gurd, University of Manchester
Brent T. Hailpern, IBM T.J. Watson Research Center
James H. Haynes, University of California, Santa Cruz
Tadao Ichikawa, Hiroshima University
Guy Johnson, Rochester Institute of Technology
Yale Patt, University of Michigan
Guylaine M. Pollock, Sandia National Laboratories
Vincent Y. Shin, MCC
Martha Sloan, Michigan Technological University
Akinori Yonezawa, Tokyo Institute of Technology

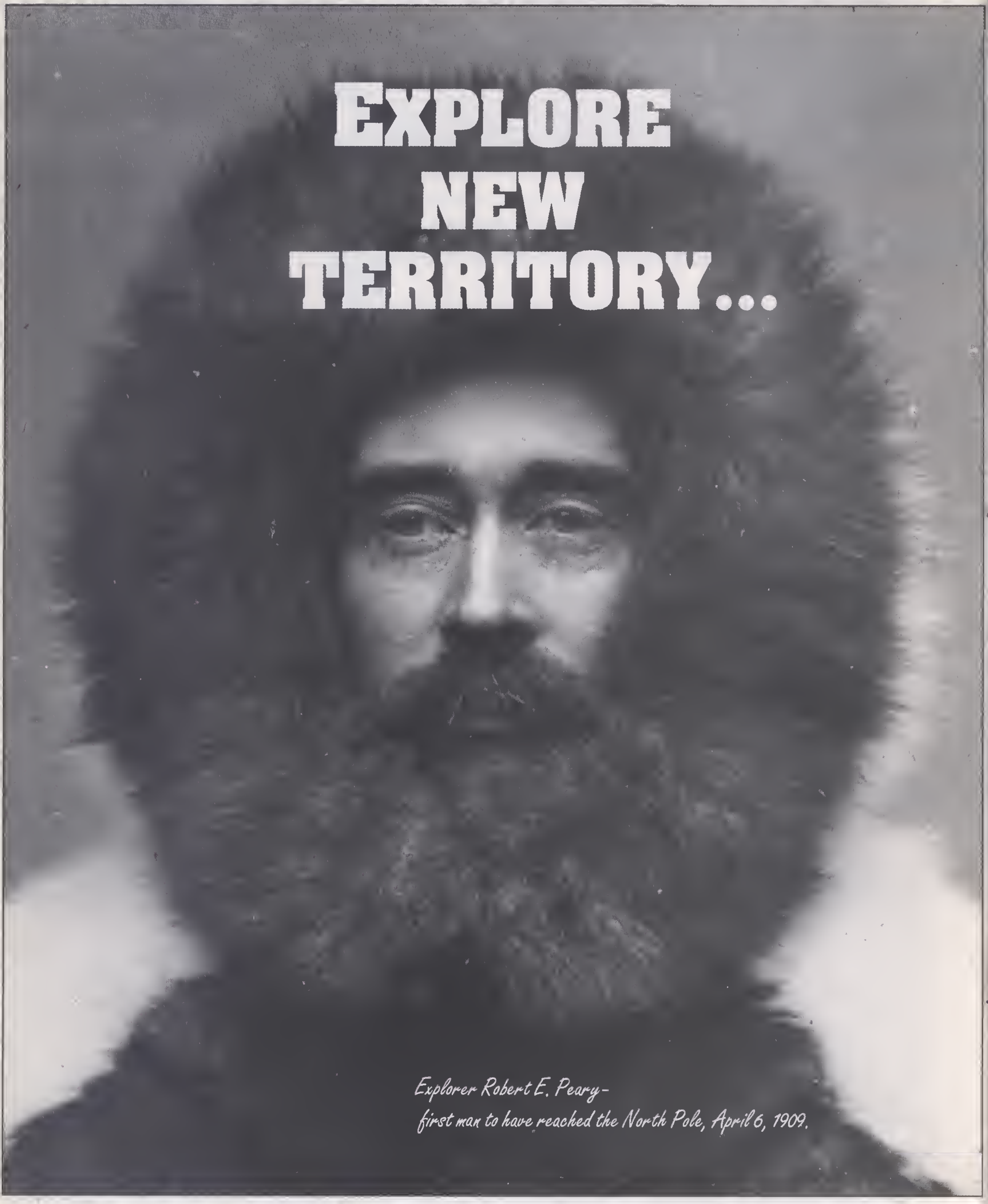
**Submissions**
Send six double-spaced copies of articles and special-issue proposals to Jon T. Butler, Dept. of Electrical and Computer Engineering, Code EC/Bu, Naval Postgraduate School, Monterey, CA 93943-5004, phone (408) 646-3299. For electronic submission, Butler's addresses are j.butler on Compmail II and butler@cs.nps.navy.mil on Internet.

**Staff**

| | |
|---|---|
| *Publisher:* | True Seaborn |
| *Editorial Director, CS Magazines:* | Marilyn Potes |
| *Assistant Publisher:* | Douglas L. Combs |
| *Assistant to the Publisher:* | Pat Paulsen |
| *Managing Editor:* | Marilyn Potes |
| *Staff Editors:* | Chuck Governale |
| | Christine Miller |
| | Bob Carlson |
| | Nancy L. Carollo |
| *Contributing Editor:* | Ware Myers |
| *Art Director:* | Jay Simpson |
| *Design/Production:* | Toni Van Buskirk, Janice Garton |
| *Membership/Circulation:* | Christina Champion |
| *Advertising Coordinators:* | Heidi Rex, Marian Tibayan |

**Magazine Advisory Committee**
Pradip K. Srimani (chair), Jon T. Butler, B. Chandrasekaran, Manuel D'Abreu, James Farrell III, Joe Hootman, Anil Jain, Sushil Jajodia, Ted Lewis, True Seaborn, Bruce D. Shriver, John Staudhammer

**Publications Board**
Sallie Sheppard (chair), Dharma P. Agrawal, Victor Basili, P. Bruce Berra, J. Richard Burke, Jon T. Butler, J.T. Cain, B. Chandrasekaran, David Choy, James Cross, Manuel D'Abreu, James J. Farrell III, Joe Hootman, Sushil Jajodia, Glen G. Langdon, Ted Lewis, Ming T. Liu, C.V. Ramamoorthy, True Seaborn, Bruce D. Shriver, John Staudhammer, Harold Stone, Steven L. Tanimoto, Joseph Urban, Ben Wah, Ron Williams



*Explorer Robert E. Peary–
first man to have reached the North Pole, April 6, 1909.*

UPI/BETTMANN NEWSPHOTOS

Microsoft Windows and Excel are trademarks of Microsoft Corp. Lotus 1-2-3 is a trademark of Lotus Development Corp. dBase II is a trademark of Ashton-Tate.

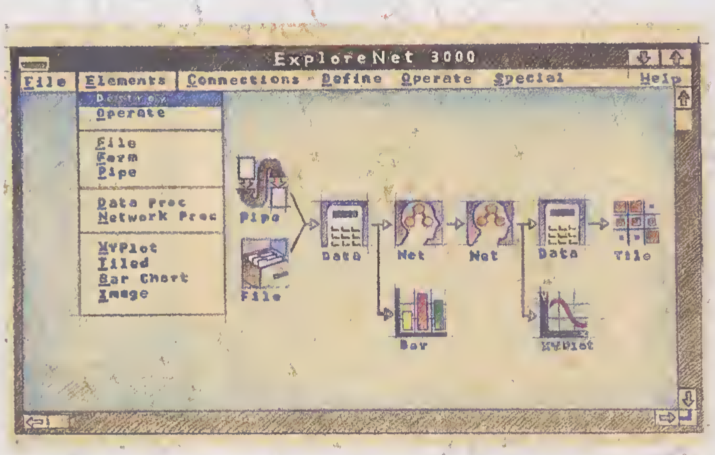
Order Now
Save over $1000 on Training and Support

# Find Hidden Answers.

## EXPLORENET 3000™

is advanced neural network software for the PC that helps you find hidden answers in your data. You will experience the ease of developing your application within Microsoft Windows™ 3.0 environment (compatible with over 3,000 other programs on the PC/AT). And best of all, you can develop applications without programming! ExploreNet 3000 applications are 100% compatible with HNC's ANZA Plus™ coprocessors.

EXPLORENET™
3·0·0·0



**File Module**

Support fixed and variable length ASCII and binary files, which can be generated by 99% of all programs. For instance, output from such popular programs as Lotus 1-2-3™, dBase II™ and Excel™ can be used without modification.

**Data Transformation Module**

Perform pre-and post-processing operations by combining functions from a library of over 50 routines. Supports scalar, vector and matrix operations.

**Pipe Module**

Receive input from and output to thousands of other programs including expert systems, databases and applications programs running on the PC under MS-Windows.

**Display Modules**

Watch your network train and learn by monitoring the results using six different display options: Plots, Tiles, Images, Forms and Charts (pie charts and bar charts).

**Network Module**

Tailor any of the 18 well-known neural networks to your application, or modify the architecture of one of the three experimental networks to your specifications.

ExploreNet 3000 will get you up and running faster than any other neural network product on the market—we'll back it with a 30-day money-back guarantee.

**Call 1-800-HNC-EXPR**

HNC™

5501 Oberlin Drive
San Diego, CA 92121
(619) 546-8877
Fax (619) 452-6524

Reader Service Number 1

Computer Society President's MESSAGE

# The future looks even better

**The Computer Society's 1990 president, Helen M. Wood, encourages continued communication between society officers and the members they serve.**

This is a natural time for me to reflect upon my year as Computer Society president — a year that, in many respects, seemed to fly by all too quickly. I have learned a number of things that I would like to share with you about the society and its relationship to its members, other parts of the IEEE, and organizations outside the IEEE family.

**Volunteers.** The Computer Society, like any association of technical professionals, encompasses at least as many ideas and opinions as there are volunteers involved. At times, this diversity stimulates some highly energetic discussions — to say the least. But one thing is clear: Those who dedicate their time and efforts in support of the Computer Society's activities truly care about the quality of the results. The society is blessed with an extraordinary cadre of volunteer leaders — hundreds of them — whose hard work makes the society the leader in its field.

**Staff.** During this year, I visited each of our four offices — in Belgium, Japan, California, and back home (for me) in Washington, DC. Regardless of the location, all of our offices have one thing in common: a staff of highly competent, dedicated professionals. Our staff provide the fulcrum on which we leverage our volunteer effort, yielding many more quality programs and services for members than is typical for organizations of our budgetary and staff size.

**Partnerships.** Back in January, I pledged to work to expand our partnerships with other professional and technical organizations — both in terms of establishing new affiliate relationships and organizing joint activities. Throughout the year, I have met and corresponded with representatives from a number of other societies. I believe we should strive to maintain active communications and to expand technical collaboration with our sister societies around the world. Such partnerships will help strengthen our ability to support the needs of computing professionals.

**IEEE.** During the past year, I have had occasion to work closely with a number of officers, committees, and boards in IEEE. We worked together to try to strengthen the institute while balancing the sometimes conflicting requirements of different parts of the whole. Some tension is inevitable in any family, but we remain a vital family. The Computer Society's relationship with the institute is important to the health of both. It should be maintained and improved for the benefit of all our members, and as I assume the new duties of IEEE Division VIII director, to which you have just elected me, I will continue my efforts to do so.

**Meeting members' needs.** By the end of 1990, Computer Society membership will approach 110,000. Our growth has been rapid, reflecting the impressive growth of the computing industry. And like the industry, we are maturing. Next year, we will celebrate the 40th anniversary of the founding of the Computer Society. It is time for us to stand back and reassess our programs. We can see from the magazines you order and the conferences you attend that your interests and priorities are changing. To meet your needs, we must make changes in the programs and services we offer, and to do this we need more direct feedback from you. Many of you have contacted me throughout the year with your comments, suggestions, and even your complaints. You have let me know what you think is important and what is not. For this I thank you and encourage you to continue communicating with all the society's leaders.

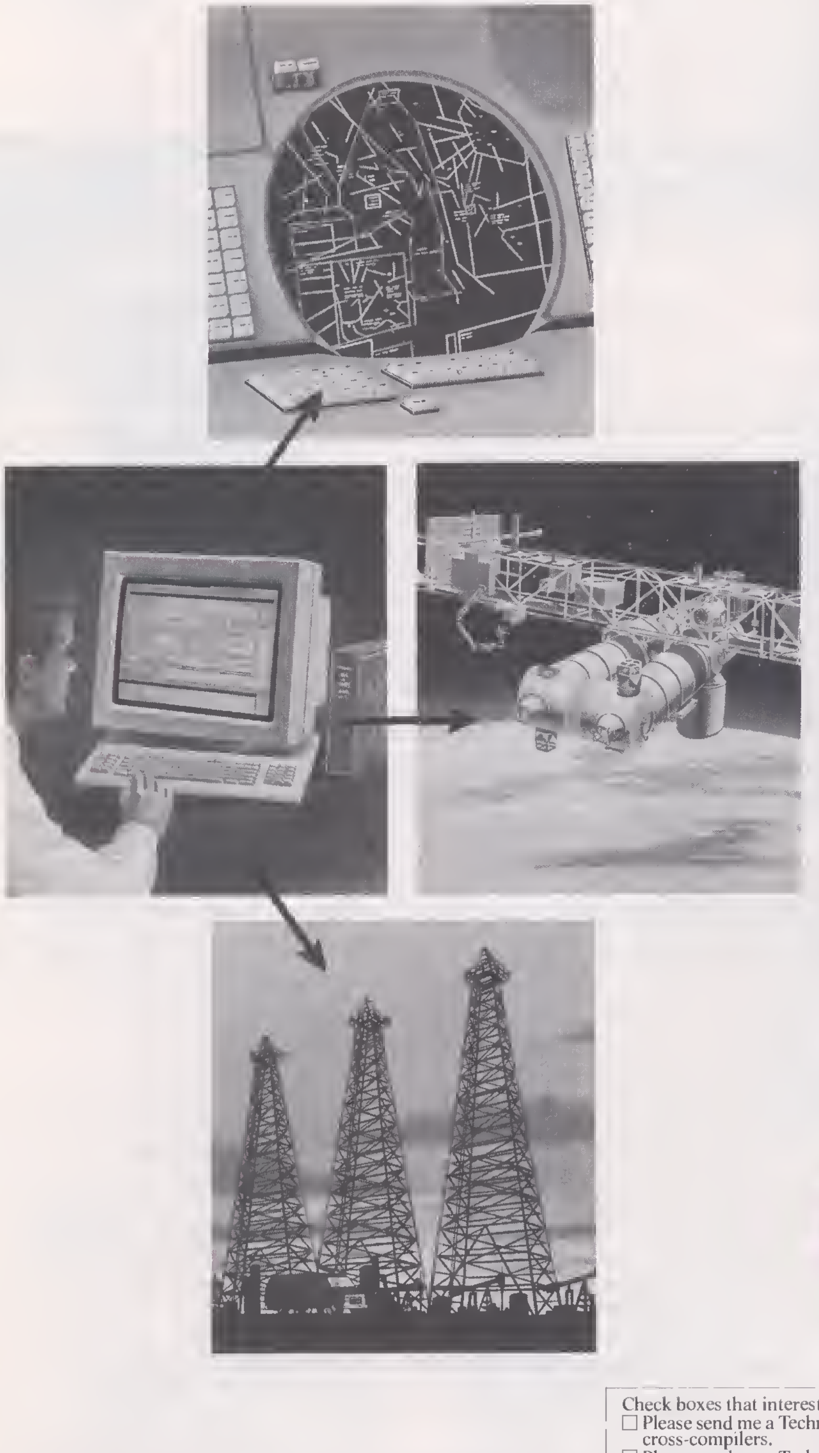
The officers and Board of Governors members are listed in every issue of *Computer* magazine and in almost all issues of our other periodicals. If you don't know their addresses, just write to them in care of any of our offices and your comments will be forwarded. If something is wrong, we need to know it. If you have a good idea about something we should be doing — a new product or service — let us hear from you. Frankly, it is easy for us as society leaders to sometimes get so caught up in running the society's business that we may momentarily lose sight of our primary objective: member service. Direct contact with members is what prevents that from happening. Praise is always welcome, criticism will be treated constructively, and new ideas are received with enthusiasm.

This column is my last chance to publicly express my appreciation to the society's officers, Board of Governors, and staff for their efforts throughout my year as president. In particular, I would like to recognize several individuals whose support has been enormously helpful. Whenever I felt the need for a broader perspective, Ed Parrish, Tom Cain, Roy Russo, and Duncan Lawrie were always willing to share their thoughts and time. Their experience, wisdom, and dedication to the society are an invaluable resource. Also, I would be remiss if I failed to recognize Michael Elliott, the society's executive director. He has been a trusted advisor and colleague.

Next month, Duncan Lawrie will succeed me as Computer Society president. I know Duncan shares my commitment to helping the Computer Society continue to serve the members and the computing profession. Through the efforts of those whom I have followed, and those, like Duncan, who will follow me, I know the society will remain healthy and strong. And with the active involvement of the membership, the future won't just be the way it used to be — it will be even better!

Helen M. Wood
Computer Society president

# Alsys Ada cross-compilers get you there in no time.



It's time you knew that Alsys, the premier Ada company, offers a range of powerful and flexible cross-compilers for all microprocessors in the Motorola MC680X0 and Intel i80X86 families* to get your applications up and running fast.

Part of the reason for this speed are powerful development tools such as AdaProbe, a source level debugger and program viewer providing facilities to address both the execution properties of a program and its static structure. In addition, there's support for placing program components into ROM, and the Alsys Multi-Library Environment allowing program units to be shared among libraries for team programming.

With Alsys' full line of cross-compilers you'll discover impressive flexibility and power. There's a configurable run-time system giving you full control over tasks, interrupts and all components of your application. The debugger and transfer utility are configurable. Best of all, it's easy to take advantage of all this power because there are only a few files to modify.

When you need to get from here to there without getting lost somewhere in between, use a cross-compiler that knows the shortest route.

alsys
Ada Now.

*The 680X0 cross-compilers are hosted on VAX, SUN 3, Apollo, HP 9000/300, IBM PS/2, PC AT, and Compaq 386. The Intel cross-compiler is hosted on the IBM PC AT and Compaq 386.

In the US: Alsys, 67 South Bedford Street, Burlington, MA 01803-5152, Tel: (617) 270-0030.

In the UK: Alsys, Partridge House, Newtown Road, Henley-on-Thames, Oxon RG9 1EN Tel: 44(491)579090

Elsewhere: Alsys, 29 Avenue de Versailles, 78170 La Celle St. Cloud, France Tel: 33(1) 3918.12.44

Check boxes that interest you. IEC12/90

☐ Please send me a Technical Summary on the Motorola MC680X0 cross-compilers.
☐ Please send me a Technical Summary on the Intel i80X86 cross-compilers.
☐ Please send me a copy of your "Many Facets of Quality" brochure.

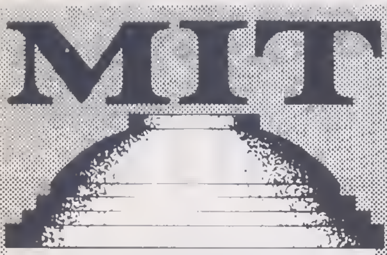
Name ______

Address ______

City ______ State ______ Zip ______

Phone (____) ______

Mail to: Alsys, Inc., 67 South Bedford Street, Burlington, MA 01803-5152



**Systems Programmer**
**Laboratory for Computer Science**

To maintain and improve a large software system written in Common Lisp. The focus will be to modify and improve the existing system.

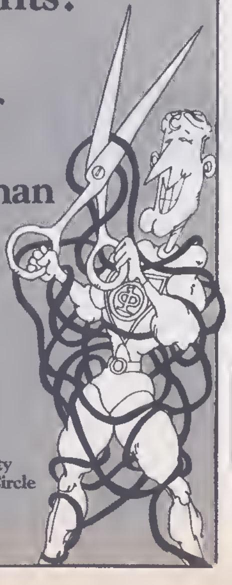
**Requirements:** an MS in Computer Science or a BS with a minimum of 3 years' experience. Expertise in Common Lisp required. Familiarity with Common Lisp on a variety of platforms, with C and UNIX and with functional languages desirable.

Interested candidates should submit two copies of a resume and cover letter referencing **Job No. R90-190** to: **James H. McCarthy, MIT Personnel Office, Bldg. E19-239, 400 Main Street, Cambridge, MA 02139.**

MIT is an Equal Opportunity/
Affirmative Action Employer
MIT is a non-smoking environment

**Massachusetts Institute of Technology**

**Late Magazines?**
**No Magazines?**
**Membership Status Problems?**
**No Answers To Your Complaints?**
**Let your Computer Society Ombudsman cut through the red tape for you.**



Ombudsman
IEEE Computer Society
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA
90720-1264

Editor-in-Chief's MESSAGE

## Withdrawal Symptoms

When you've enjoyed working with talented, creative people for eight years, saying good-bye can be a tough thing to do. Such are the circumstances that I find myself in now — finishing a four-year tenure as editor-in-chief of *Computer*, immediately after four years as editor-in-chief of the then start-up magazine, *IEEE Software*. After eight years of working a part of every week with the technical editorial content of these two magazines, I cannot help but muster up thoughts of impending withdrawal symptoms. I face a future of not being directly and intimately involved with the myriad of activities that comprise the "job" of an EIC — working with authors, referees, guest editors, department editors, and people with and without vested interests; debating issues and setting policy at the Magazine Advisory Committee; fighting for page count and budget at the Publications Board meetings; and on and on.

I entertain a certain amount of pride as I look back on the birthing (*IEEE Software*) and technical growth (*Computer*) that was accomplished during this time. It obviously could not have been done without the incredible dedication of a host of editorial board members, department editors, and guest editors; a legion of harassed authors and referees (who undoubtedly heard the word "quality" from me one too many times); and the significant creative strengths and professional work of the Computer Society's publications staff. All this was done, of course, under the normal intensity of deadline rush that permeates the publishing industry.

For the counsel, guidance, and insight they have given me over these eight years, I must thank True Seaborn and Marilyn Potes of the Publications Office; editorial board members (at one time or another) Dennis Allison, Ted Lewis, Richard Eckhouse, Edmund Gallizzi, Yale Patt, Michael Evangelist, and Wiley McKenzie; able advisers Roy Russo, Oscar Garcia, Tom Cain, James Aylor, and Ronald Hoelzeman; Computer Society executive director T. Michael Elliott; and trusted friend and mentor Harold Stone. Among the numerous referees who did an exceptional job in refereeing manuscripts is the new editor-in-chief of *Computer*, Jon Butler of the Naval Postgraduate School.

Lastly, I must publicly thank the three institutions that supported my efforts during this time: the University of Southwestern Louisiana (three years), the University of Hawaii (one year), and IBM's T.J. Watson Research Center (four years). I had the benefit of talented assistants at each institution: Pat Rees and Stephanie Denton at USL, Stacy Hunt at UH, and Faith Compo at IBM. They all added value to and took pride in what they were doing. I sincerely thank the Computer Society for giving me this opportunity to do something that was thoroughly enjoyable for such a long period of time.

Bruce D. Shriver
Editor-in-chief

IEEE

# ASIC'91

Fourth Annual IEEE International

# ASIC Conference and Exhibit

(formerly ASIC Seminar and Exhibit)

**September 23-27, 1991**

Rochester Riverside Convention Center
Rochester, New York 14604

Sponsored by the IEEE Rochester Section in cooperation with the IEEE Computer Society

## *CALL FOR PAPERS, WORKSHOPS, and TUTORIALS*

The International ASIC Conference and Exhibit is organized to promote the practice of **ASIC engineering** by providing ASIC and systems level engineers/scientists and managers with knowledge of the tools and techniques required in all phases of ASIC design and implementation. It emphasizes the understanding of practical issues, technical details, tradeoffs and economics of system integration using **standard cell, gate array, programmable array, cell compiler, and full custom techniques in both the digital and analog domains.** The conference offers:
(1) An in-depth introduction to ASIC implementation for the **systems engineer**, (2) An advanced program for the **experienced ASIC practitioner**, (3) A forum for ASIC **users and vendors** to share case design experiences, and (4) **Executive overviews** of ASIC trends, strategy, economics and competitiveness.

***Proposals to organize Workshops, Tutorials, or Sessions are invited.***

**Technical papers** to cover ASIC applications in the following areas are solicited.

* **ASIC Design & Applications:** HDTV systems, Medical electronics, Auto ASICs, Communication ASICs, Image processing, Electro-optics Interface
* **ASIC CAD Tools and Simulation:** Schematic capture, VHDL, Logic & timing synthesis, Simulation models, Cell libraries, Performance evaluation.
* **ASIC Testability & manufacturing:** Circuit testing, Reliability, Fault tolerance, ATPG, Multichip modules, and Packaging technology.
* **Economics and Management of ASIC Projects:** Cost analysis & comparison, Benchmarking, Marketing, Scheduling, Technology impact.
* **Mixed Signal ASIC Design:** Mixed analog/digital design, testing, modeling, interfaces, and simulation.
* **ASIC Education:** Training programs in VHDL, Simulation, CAD tools, Course development in both industry and universities.
* **Programmable Logic Devices:** FPGA, PLA, PAL, Technology and applications.

Workshops (Sep 23, 24) Four or eight hour technical workshops covering ASIC design knowledge and skills. Proposals to form these workshops for either introductory or advanced level are invited. ASIC industry, as well as universities are encouraged to submit proposals for consideration. Contact the Workshop Chairman:
Dr. Lionel J. D'Luna, MC-02036, Eastman Kodak Company, Rochester, NY 14650. Phone: (716) 477-8386. Fax: (716) 477-4947

**Tutorials:** Proposals are solicited for one or two hour educational tutorials covering ASIC fundamentals, mixed signal design, timing and logic synthesis, design for testability, VHDL, managing ASICs, and manufacturing considerations. Contact the Tutorial Chairman:
Glen W. Brown, MC-02015, Eastman Kodak Company, Rochester, NY 14650. Phone: (716)722-4755. Fax: (716) 477-4947

## INSTRUCTIONS TO AUTHORS:

Authors for papers, tutorials, and workshops are asked to submit six copies of a review package, which includes a 500 word summary and a 50 word abstract, typed single-spaced on a 8-1/2 x 11 white paper for paper selection purposes. Author's name, affiliation, complete mailing address, telephone number and telex/fax MUST appear on the summary. The summary should clearly state: 1) Title of the paper; 2) The purpose of this work; 3) The major contributions to the art; 4) The specific results and their significance; and 5) Technical area.

## HIGHLIGHTS

* Workshops (2 days)
* Tutorial Sessions
* Technical Papers
* Technical Exhibits
* Proceedings of Papers/Tutorials
* Evening Panel Discussion
* Spouse Program

**Send Technical Proposals to:**
Lynne M. Engelbrecht
ASIC Conference Coordinator
170 Mt. Read Blvd.
Rochester, NY 14611
Phone: (716)328-2310
Fax: (716)436-9370

* Summaries and proposals deadline: **March 1, 1991.**
* Notification of Acceptance: **April 15, 1991.**
* Final Camera Ready Manuscript by **June 10, 1991,**

Further information may be obtained from:

| Conference Chairman | Technical Program Chairman | Exhibit Chairman |
|---|---|---|
| Dr. Kenneth W. Hsu | Dr. Y. Tim Tsai | Peter D. Parslow |
| Rochester Institute of Technology | Eastman Kodak Company | P.R. Communications |
| Computer Engineering Department | MS 02015 | 469 Blossom Road |
| Rochester, NY 14623 | Rochester, New York 14650 | Rochester, New York 14610 |
| Phone: (716) 475 - 2655 | Phone: (716) 722 - 4896 | Phone: (716) 288-7900 |
| Fax: (716) 475 - 6879 | Fax: (716) 477 - 4947 | Fax: (716) 288-7909 |

COMP

SPONSORED BY THE
IEEE COMPUTER SOCIETY

*Compcon Is The Longest, Established Computer Conference Providing A Complete Update Of The Most Timely Trends And Developments In Computing.*

Technological Updates Provide Personal Insights From Industry Leaders In A One-hour Format

Technical Sessions Provide Presentations In A 90-minute Format

Tutorials Provide In-depth Technical Presentations In An All-day Format

*Some Conference Highlights And Notable Topics*

- Trends In UNIX Software
- Desktop SPARC Systems
- Towards The Single Chip PC
- Protection For And Against Information
- Intergraph SuperScalar Clipper
- 100 MIPS Processors
- The MPEG Standard
- Survey Of Computing In Japan
- Parallel Computing
- Next Generation H-P RISC Workstations
- DECstation 5100
- SQL Access: DataBase Interoperability
- New SIMDArchitectures
- Handwriting And Text Recognitio
- Multi-Media
- Object Oriented Technology
- Video Processors & Multi-Media

# spring compcon91

THE INSTITUTE OF ELECTRICAL AND ELECTRONIC ENGINEERS, INC.

## *Technological Updates From Industry Leaders*

Towards $10^{15}$ MIPS
*By Eric Drexler*

America's Answer To Foreign Competition: The Entrepeneur And Inventor
*By Gilbert Hyatt*

Virtual Reality: A Computer Science Perspective
*By Jaron Lanier*

Televisions Of Tomorrow: Signals With A Sense About Themselves
*By Andy Lipman*

Role Of GaAs In Digital Design
*By Lou Tomasetto*

## *Special All Day Tutorials*

Computer Architecture Choices
*By Yale Patt*

MACH Distributed Operating System
*By David Black*

Fundamentals Of X Windows & User Interface
*By Chuck Clanton*

Case Tools For Requirements Analysis & Software Design
*By John Brackett*

The Future Of Supercomputing
*By Stephen Lundstrom*

Transaction Processing Concepts & Techniques
*By Jim Gray*

VHDL- Hardware Design Language
*By Stan Mazor*

## *Registration Information*

*When:* February 25 through March 1, 1991

*Where:* Cathedral Hill Hotel, Van Ness at Geary, San Francisco
*For Hotel Reservations:* (415) 776-8200 Mention COMPCON 91

*For Conference Registration* (415) 423-3490 Ask For COMPCON
*Or Send Email To:* compcon91@lbl.gov

| | | | |
|---|---|---|---|
| *Prices:* | Conference: | Members | $250 by Feb. 8, $285 after Feb. 8 |
| | | Non-Members | $325 by Feb. 8, $360 after Feb. 8 |
| | Tutorials: | Members | $250 by Feb. 8, $285 after Feb. 8 |
| | (each) | Non-Members | $325 by Feb. 8, $360 after Feb. 8 |
| | One Day Conference Rate: (at conference only) | | $150 Members<br>$210 Non-Members |

*See The January 1991 Issue Of Computer For The Advance Program*

*See The January 24th Issue Of Electronic Design For Conference Features*

Do you practice Yourdon/DeMarco Structured Analysis?
Do you need to produce high quality DFD's?
Do you need a tool that fits your budget?

## Then MacBubbles™ was made for you!

**MacBubbles** supports the process of structured analysis with:

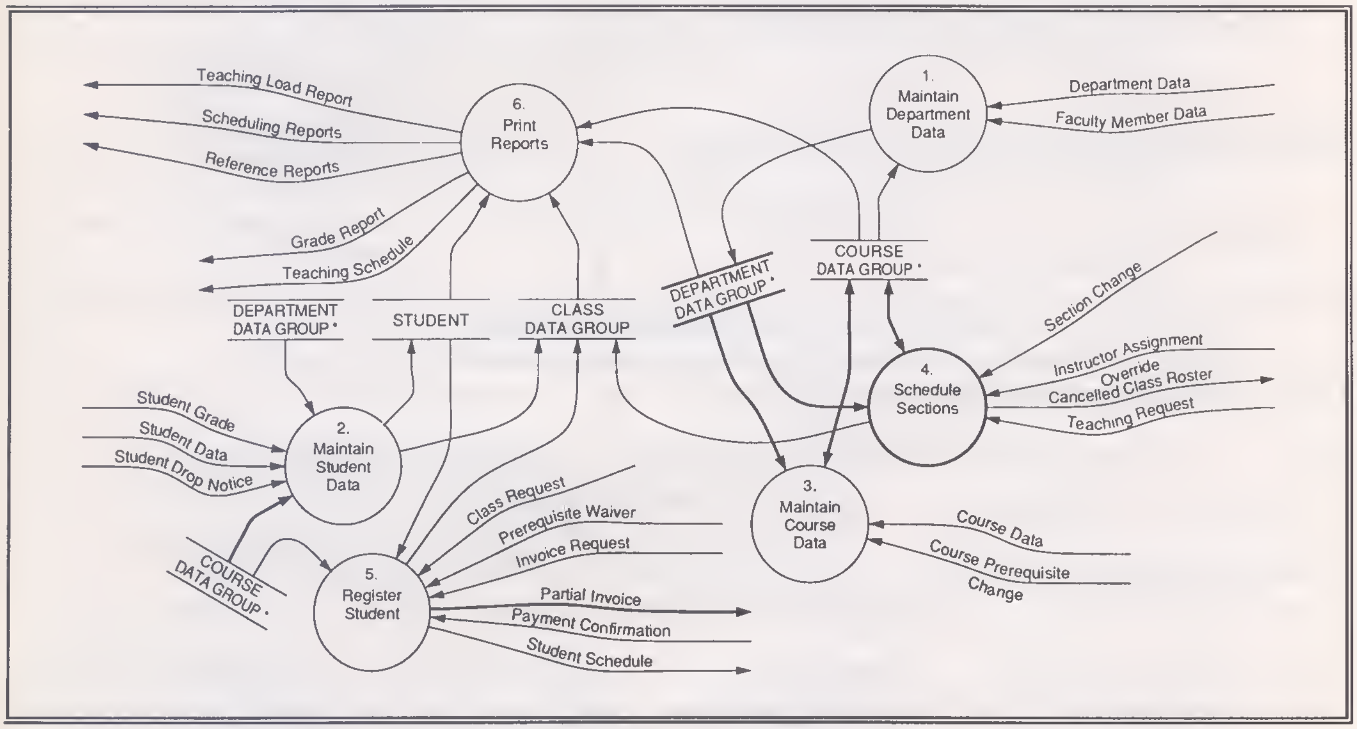
- Facilities for creating, modifying, leveling and expanding DFD's
- A Data Dictionary that maintains and lists full where-used information
- Automated balancing checks for DFD's and minispecs

**MacBubbles** improves productivity:

- Easy to learn
- Easy to use
- Responsive

**MacBubbles** is economical:

| | |
|---|---|
| Single copy price | $779.99 |
| Multi-copy discount available | |
| Demo disk | $25.00 |



Sample Data Flow Diagram

**MacBubbles** requires:

- A Macintosh Plus or SE
- Two floppy drives or a hard disk
- A LaserWriter for high quality output

*StarSys, Inc.*
11113 Norlee Drive
Silver Spring, MD 20902-3619
(301) 946-0522

Reader Service Number 3

# Do Parallel Languages Respond to the Needs of Scientific Programmers?

Cherri M. Pancake, Auburn University
Donna Bergmark, Cornell University

**Scientific researchers don't develop parallel programs the way computer scientists do. This article explains why existing languages may not provide enough support for scientific programming.**

The interest of computer scientists in parallel programming began in the area of operating systems, where program segments executed independently in real or simulated parallelism. As parallel technology evolved, new research efforts were devoted to exploring the effects of nondeterminism in computational systems. Most current research in parallel programming concerns techniques for specifying and controlling concurrency. From the modeling of networks to the development of parallel algorithms, concurrency is an integral feature of program development and is taken into account from the earliest stages of design.

Computational scientists approach parallel programming in a different way. Although the physical world they model is inherently parallel, scientific programmers have become accustomed to using sequential techniques for its study. Their interest in parallelism evolved from the desire to improve the performance of sequential algorithms applied to large-scale numerical computations. These users want to take advantage of the computational power provided by multiple processors, not the effects of concurrency. They view nondeterminacy as an undesirable side effect rather than a property to be explored. Instead of integrating parallelism into the design process, they incorporate it after the fact to speed up applications that were tested and debugged in a sequential environment.

Scientific researchers seem to view the future of parallel computing with optimism, as demonstrated by their enthusiastic response to the increasing availability of parallel facilities. At the Cornell National Supercomputer Facility, for example, parallel processing capability tripled during the past three years, while parallel processing usage (in terms of CPU hours) increased 22 times. Nevertheless, researchers estimated that only one in 20 user programs executed on the facility's 12-processor supercomputer complex is parallel.[1]

User surveys indicate that many more applications could take advantage of multiprocessing capabilities were it not for the difficulty of reformulating sequential code. Parallelizing compilers offer a fast and convenient way to incorporate concurrency, but the speedups achieved by automatic transformations alone are disappointing. Hand-coded parallelism, on the other hand, is both difficult and time-consuming. As a result, parallelism remains inaccessible or underutilized by most of the user community.

It is not clear how easily or effectively parallel techniques can be integrated into the scientific programming process. Critical concerns include the extent to which existing sequential programs can be converted to parallel form, how successfully the results can be ported to other systems, and how easily program components can be reused in building new applications.

This article considers parallel programming from the viewpoint of scientific researchers, focusing on their requirements for language support and considering a number of questions. How do scientists go about developing parallel applications?

0018-9162/90/1200-0013$01.00 © 1990 IEEE

What role does language play in determining the success of their programming efforts? How much should scientific programmers be expected to know about parallel languages and machines? What can computer scientists do to facilitate parallel scientific programming?

The discussion centers on the "mainline" supercomputers for scientific and engineering applications:[2] vector and scalar MIMD (multiple instruction, multiple data) multiprocessors. Other high-performance architectures — notably SIMD (single instruction, multiple data) — have gained popularity with certain segments of the scientific community, but their use is not as widespread and the language support is not as varied.

Most parallel scientific computing is still carried out on general-purpose computers at facilities that are subsidized by federal or state agencies (such as the National Science Foundation's supercomputing centers, state supercomputing centers, and the national laboratories). What's more, their long history means MIMD systems and the associated software have attained a higher level of product maturity. Although the languages and tools developed for SIMD and other new architectures may prove to be ideal for many scientific applications, the issues of software support are still emerging.

## Programming as problem solving

Scientists employ computation for solving problems related to the physical world. Like other tools, the computer imposes restrictions on the way the problem is formulated for solution and the types of solutions that are possible. We cannot appreciate the difficulties confronting the scientific programmer without understanding the problem-solving process itself. Program development involves a series of steps:

- delineation of the problem domain and selection of a problem-solving strategy,
- formulation of an algorithmic solution,
- implementation using a programming language,
- translation of the program into executable form, and
- program execution.

This problem-solving system can be characterized as four subsystems operating at different levels of abstraction (Figure 1). Each subsystem defines its own collection of objects, a set of operations or manipulations applicable to those objects, and domains representing the values each object may assume.

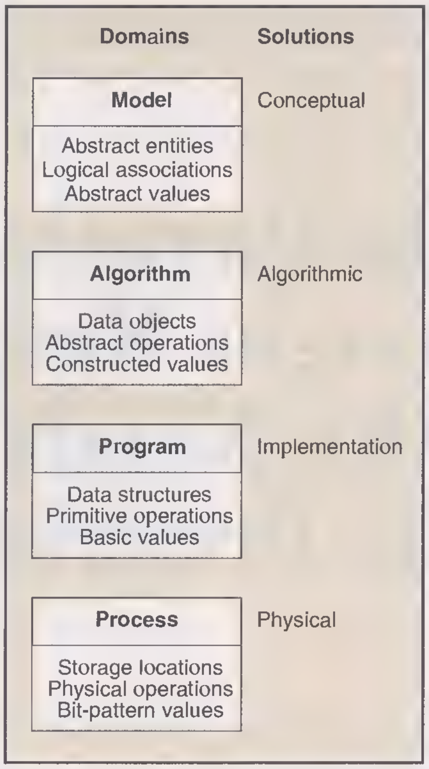


**Figure 1. Problem-solution system.**

**Conceptual level.** The *conceptual solution* occupies the highest level in Figure 1. Here, the problem is expressed in the abstract terms of human reasoning and our perception of the laws governing nature. The solution can be portrayed in terms of abstract entities, logical association among entities, and the attribution of abstract values. The problem-solving strategy is outlined in very general terms, without regard to the capabilities of the system on which it will be implemented. The description is usually in the form of natural language text or diagrams.

Consider, for example, an application of dynamic programming to model the orbital control of satellites. The satellite is equipped with tiny motor devices that must be fired in strict coordination to effect a change in direction or positioning. The control to be exerted is nonlinear. This problem subdivides naturally into two phases: a "forward sweep" to simulate the application of some control policy, and a "backward sweep" to calculate the effects of the policy, using cost/gain analysis to select a successor policy. Processing will continue iteratively, with each successive policy more refined, until some threshold of optimality has been reached.

The diagram of Figure 2 represents a conceptual solution. The problem entities are abstracted collections of data, associated logically in terms of the control policy that will be applied and refined in successive iterations. *A* through *E* are collections of values, representing systems of equations that must be solved to determine the effects of the control policy.

It is at this level that the programmer determines which portions of the solution are candidates for parallelization. Since a scientific user has speedup in mind, attention is focused on computationally intensive activities. Whenever the activities are logically independent (for example, transformations on *B, C, D,* and *E* at the start of the backward sweep), they can be grouped and marked for simultaneous execution. Similarly, when a collection of calculations is to be applied to independent data subsets (for example, the plane-wise transformations of *A*), the instructions can be replicated across multiple processors.

Although the degree of parallelism is limited by the number of physical processors available, this is not reflected in most conceptual solutions. Instead, the programmer identifies the maximum degree of parallelism that seems "natural," given the logical constraints of the problem. Figure 2 therefore shows execution streams (the computational boxes) that dynamically vary in number from one to four. No attempt is made to indicate what should be done with processors during the periods when they are not needed, nor how the solution should be altered if fewer than four processors are available.

Scientific users know that parallel execution can incur a substantial amount of hidden overhead. The special considerations involved in determining whether or not parallelization is warranted for a particular task on a particular machine, however, are beyond the experience of most applications programmers. (Compare the examples in references 3, 4, and 5.) These programmers are likely to assume that any sufficiently time-consuming activity that meets the criterion of computational independence can be parallelized effectively.

**Algorithmic level.** Below the conceptual level in Figure 1 is the *algorithmic solution*, which defines the specific steps that are required to solve the problem.

Although the operations may still be somewhat abstract, they are applied to data objects (such as a matrix) with specific ranges of values.

Although many scientific models, like the physical world they represent, are inherently parallel, the formulation of algorithmic solutions typically involves a sequential approach. Steps are expressed as "find $x$ such that $y$" rather than "accomplish actions $x$, $y$, $z$ concurrently within time $t$." A notational form is chosen on the basis of appropriateness to the logical model rather than any relationship to the computing environment in which the solution will be carried out.

In our satellite example, the general algorithm — an application of dynamic programming techniques — subdivides into a series of subalgorithms. Figure 3 represents a portion of the specification for the backward sweep, where the eigenvalues of matrix $D$ are calculated using the cyclic Jacobi method. As in most scientific programs, a widely published numerical method has been employed to reduce the amount of programming effort required and to improve the reliability of the results.

The nature of the algorithmic specification reflects the fact that problem solution will be carried out on a computer. The choice of subalgorithms may also be influenced by a consideration of how many physical processors will be available and whether memory will be shared or distributed. At this level, however, the description is generally machine-independent (that is, not tied to any particular architecture or operating system).

There is generally no explicit mention of parallelism in the algorithmic solution. Although recent years have seen increasing interest in numerical methods designed to exploit multiprocessing capabilities, established techniques are still almost exclusively sequential. Consequently, each subalgorithm is described as a single sequence of steps. If parallelism occurs at this level, it is limited to the notion that two subalgorithms may be allowed to proceed concurrently.

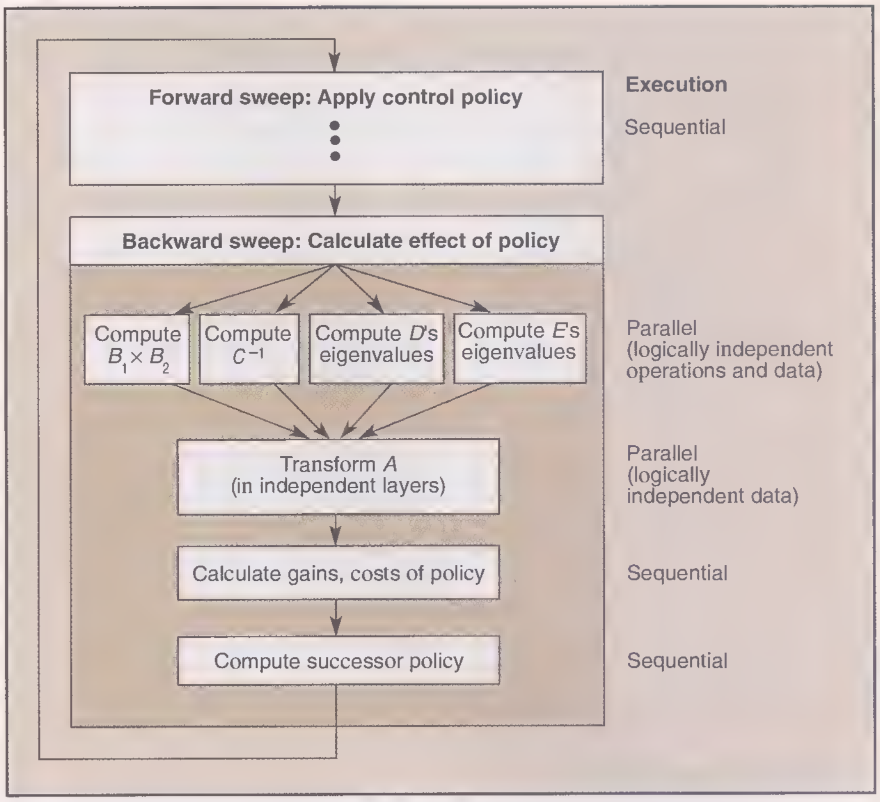

Forward sweep: Apply control policy
Backward sweep: Calculate effect of policy
Compute $B_1 \times B_2$
Compute $C^{-1}$
Compute D's eigenvalues
Compute E's eigenvalues
Transform A (in independent layers)
Calculate gains, costs of policy
Compute successor policy
Execution
Sequential
Parallel (logically independent operations and data)
Parallel (logically independent data)
Sequential
Sequential

**Figure 2. Example of conceptual solution for satellite control.**

Calculate eigenvalues of matrix $D$, using the cyclic Jacobi method to transform $D$ to diagonal form.

1. Select $d_{ij}$, the next nonzero element in a predetermined traversal of the entries above the diagonal.
2. Choose value for $\Theta$ such that $d_{ij}$ can be reduced to zero:
   a. If $d_{ii} = d_{jj}$, $\Theta = \text{sign}\ (d_{ij}) \cdot \Theta_{\text{constraint}}$
   else $\Theta = 1/2\ \arctan \cdot (2d_{ij}) / (d_{ii} - d_{jj})$.
   b. If $\Theta$ not within range, adjust: $\Theta = \Theta - \text{sign}\ (\Theta) \cdot 2\ \Theta_{\text{constraint}}$.
3. Construct the plane rotation matrix $R$.
4. Transform $D : D^{\text{next}} = R^{T}DR$.
5. Repeat from 1 until $D$ is diagonal.

**Figure 3. Algorithmic solution for one portion of the satellite example.**

**Implementation level.** Between the algorithmic and physical levels in Figure 1 is the *implementation solution*, which serves to bridge the gap between the representation of the problem as abstract manipulations and as the physical operations to perform those manipulations. Here, the problem is reexpressed in the terms supported by a programming language: data structures, primitive operations, and basic values. Since the purpose of this description is to allow an automated translation to machine code, the programming language imposes a rigorous formalism. At the same time, the language may be far removed from the functional capabilities of the physical system in order to provide expressiveness, generality, and portability.

This phase in program development is the most challenging. The programmer must devise concrete representations of all problem data and describe them using the restrictive notation of the programming language. The steps of each subalgorithm must be expanded as well; what were previously high-level equations or operations

```
        SUBROUTINE EIGEN(A,DIMEN,MAXIT,TOL,TCONSTR)
        INTEGER    ITER, MAXIT, DIMEN, ...
        LOGICAL    CONVRG
        REAL       TOL, EFFZERO, THETA, TCONSTR, ...
        REAL       A(DIMEN,DIMEN)
        PARAMETER (EFFZERO = 1E-10)
        DO 100    ITER = 1,MAXIT
            DO 110 I = 1,DIMEN-1
                CONVRG = .TRUE.
                DO 120 J = I+1,DIMEN
C Ignore next element if too small
                    IF (ABS(D(I,J)).LE.TOL) GOTO 120
                    CONVRG = .FALSE.
C Set new Theta
                    IF (ABS(D(I,I)-D(J,J)).GT.EFFZERO) THEN
                      THETA = ATAN(2*D(I,J)/(D(I,I)-D(J,J)))/2
                      IF (ABS(THETA).GT.TCONSTR) THETA =
                       THETA-SIGN(2*TCONSTR,THETA)
                    ELSE
                      THETA = SIGN(TCONSTR,D(I,J))
                    ENDIF
C Construct plane rotation matrix
                    ...
C Perform transformation
                    ...
  120           CONTINUE
  110       CONTINUE
C Check for convergence
            IF (CONVRG) RETURN
  100 CONTINUE
C Maximum iterations exceeded - activate error handler
                    ...
```

**Figure 4. Implementation solution corresponding to the algorithm of Figure 3.**

are now specified in detail via control structures and statements. Since compilers often extend the programming language to take advantage of architecture-specific features, the description may be somewhat machine-dependent. Figure 4 represents a portion of the Fortran 77 code used to implement the eigenvalue subalgorithm.

The algorithmic solution may require considerable massaging due to the requirements of computer arithmetic. To ensure that processing will terminate, series calculations must be truncated. Truncation is performed here by comparing array elements with the predetermined tolerance TOL prior to using them as the basis for transformations. As a safety measure, calculations may be forcibly terminated; here MAXIT sets a limit on the number of iterations permitted. Comparable efforts are needed to deal with round-off and cancellation errors caused by a lack of numeric precision. Finally, the stability and conditioning of each numerical method must be taken into account.

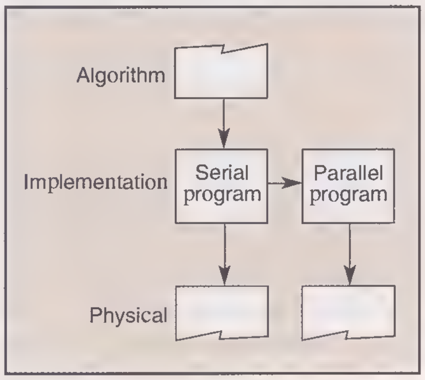


**Figure 5. Subdivision of implementation into serial and parallel phases.**

In the example of Figure 4, there has been no attempt to improve efficiency by restructuring the program code with temporary variables to eliminate array accesses. The algorithm itself could be improved to reduce computation by performing only partial matrix multiplications. Scientific programmers typically concern themselves with developing a functional solution, leaving improvement activities until later.

For similar reasons, no parallel constructs are shown, although parallelism must eventually be incorporated at this level of description. Most scientists develop sequential implementations first, adding parallel features once they are confident that the solutions work. In effect, the implementation solution subdivides into two phases: serial and parallel (Figure 5). Since the effects of the programming language are felt throughout the implementation process, we will discuss the role of the language before examining how parallelism is actually incorporated.

We do not discuss the physical level of our problem-solving system here because this level is not under the direct control of the programmer.

## The impact of language

Multiple restructurings complicate program development. The initial, abstract model for problem solution must be reformulated by the programmer as an algorithmic solution, transformed manually into program code, and translated automatically into executable code. In addition to imposing development overhead, each restructuring is a source of potential error and distortion. In particular, the effectiveness of each restructuring — and summarily, of program development as a whole — is bounded by factors related to how the mapping is accomplished (Figure 6).

The first and third transformations pose no special problems. The first, carried out at a logical level, is bounded by the programmer's ability to decompose an abstract model into a sequence of suitable, high-level representations and operations. This is the most comfortable and best understood restructuring for the scientific programmer. What's more, the increasing availability of published methods for standard numerical computations has streamlined the process. The third transformation has also benefited from past research and experience. Its effectiveness, bounded by the accuracy of the mapping from language constructs to machine instructions, relies on compiler technology that is largely beyond the control of the programmer.

The second transformation, on the other hand, poses special difficulties for the scientific researcher. Since it involves a translation from logical to quasi-physical form, its success relies on the programmer's understanding of computational methods. No automated tool can repair incorrectly stated algorithms or compensate for ill-chosen numerical techniques.

The user's level of programming expertise and experience in developing similar applications are also important. In many cases, however, the most critical factor is the programming language itself. Language provides a framework for describing how the problem's solution will be achieved. Ultimately, even the most qualified programmer must depend on language features to bridge the "semantic gap" between logical and physical problem solution.

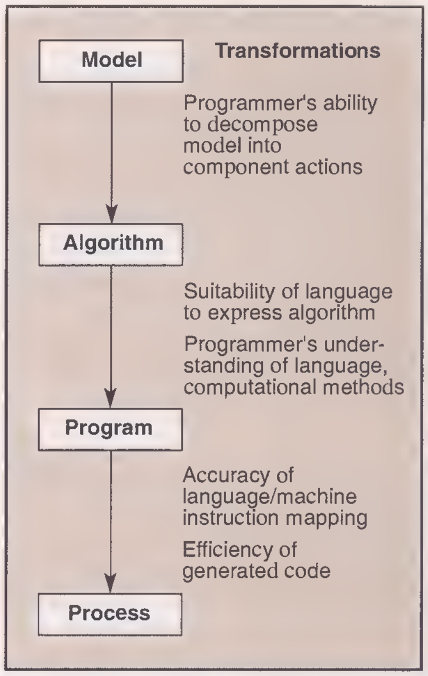
Programmers have long been aware that language design has significant impact on how easily an algorithm can be transformed into workable code. Few would elect to implement list-processing software in Fortran or computationally intensive matrix algorithms in Lisp. Even the so-called general-purpose languages are recognized as being suited to certain problem-solving approaches. It is always possible to construct an accurate implementation using an inappropriate language. The transformation process is more tedious and error-prone, however, when the conceptual models supported by the language relate only peripherally to the problem-solving model of the programmer.

More than 15 years ago, Wirth reflected that the goal of a programming language

> is to provide a framework of abstractions and structures that are appropriately adapted to our mental habits, capabilities, and limitations.... A form must be found for these facilities which is convenient to remember and intuitively clear to a programmer, and which acts as a natural guidance in the formulation of his ideas.[6]

If an appropriate high-level structure is available, users can take full advantage of the compiler's semantic checking as a safeguard against many forms of runtime errors. The use of clearly defined language structures also makes it possible to achieve an acceptable degree of independence from the underlying architecture. This approach allows the porting of programs from one system to another, as well as accommodating system upgrades or other modifications. Finally, when language syntax corresponds closely to the problem domain, program code more visibly reflects logical concepts. This facilitates debugging and improves the likelihood of reusability.

Methods for improving the effectiveness of sequential program development reflect the multilayered organization of programming activities. Formal design methodologies are used to structure the conceptualization process so the conversion to programming language constructs will be more straightforward. This is the equivalent of moving the algorithmic solution closer to the implementation. Programming language designers often take the opposite approach. By introducing more abstract language features, they elevate the implementation solution, shifting transformation responsibilities from the programmer to the compiler.


Model
Transformations
Programmer's ability to decompose model into component actions
Algorithm
Suitability of language to express algorithm
Programmer's understanding of language, computational methods
Program
Accuracy of language/machine instruction mapping
Efficiency of generated code
Process
**Figure 6. Limitations on the effectiveness of program development.**

Such fine-tuning of a programming system is possible precisely because of the clear delineation between the two levels of transformation. This property, referred to as logical independence, represents a commitment to maintaining a separation between machine-dependent and machine-independent factors. With few exceptions, today's sequential programming languages shield the programmer from implementation details. At the same time, increasing efforts are devoted to support the user's mental models through provision of such features as abstract data types and predefined collections of reusable modules.

Commercial and public-domain software libraries extend the capabilities of programming languages by offering convenient, efficient, and reliable numerical techniques for a wide range of disciplines. The lessons of three decades of sequential program development are clear: Programmer effectiveness improves when language structures are moved away from physical issues and toward logical models.[6,7]

## Language support for parallelism

The introduction of parallelism complicates the physical end of program implementation. The programmer is now concerned not just with data objects and operations, but also with the interaction and relative timing of independent entities. What activities can be carried out in parallel? What data must be shared among them? Experienced programmers realize that effective parallelization involves other issues as well. How can the work load be distributed to minimize processor idle time? How can processor activities be coordinated? How can the correctness of data values be ensured when the order of access is unpredictable?

Once again, the programmer must rely on the programming language to describe how parallelism is to be incorporated, but now language support can be implicit as well as explicit. Parallelization is implicit when the compiler can recognize potentially concurrent portions of a sequential program and generate parallel code. Implicit parallelization requires extensive analysis of the dependencies among data items and cannot guarantee an optimal solution. Although parallelizing compilers have been cited as a promising direction for the future, the versions currently in production have limited capabilities. Recent studies challenge the usefulness of this approach.[7-9]

Parallelism becomes explicit when the programmer must specify the nature and extent of concurrent activities through language constructs. Three mechanisms have been used to support parallel capabilities:

- incorporating parallel features as integral parts of a language's design,
- adding parallel extensions to an existing sequential language, and
- providing high-level interfaces to parallel routines stored in a system library.

**Concurrent languages.** Integrating parallelism directly into the design of a concurrent programming language offers the best chance for clear and unified support of conceptual models. This approach maximizes the potential for automatic error detection and facilitates the development of effective debugging tools. Occam, Ada, Concurrent Pascal, and Parlog are examples of languages in which a significant number of structures are devoted to supporting parallelism. Other languages such as PL/I, Mesa, and Algol68 integrate parallel features in the original language definition, but on a much simpler scale.

Unfortunately, experience has shown that it is extremely difficult to design features that are both generally applicable and easy to use. Furthermore, the costs of developing applications are not limited to the acquisition of a suitable compiler. The programmer must learn a new philosophy of program development as well as a new language structure. Any existing code segments must be reformulated and recompiled.

**Language extensions.** Sequential languages can be extended to handle parallelism by the addition of compiler directives or macros. This approach clearly facilitates the parallelization of "dusty deck" programs (codes that have been in use so long that there is little or no documentation on how they were developed). The learning curve is also much better, since the programmer need only assimilate a few structures and identify the situations when they are appropriate. Parallel Pascal, Concurrent C, Multilisp, and most of the parallel Fortrans fit into this category. The primary disadvantage is that it is extremely difficult to integrate parallel constructs cleanly and logically.[5,10]

When the extensions are implemented in the form of macros handled by a preprocessor, many of the compiler's error-checking capabilities must be sacrificed. Other problems include a strong machine- and dialect-dependence, resulting in decreased portability. Extensions may also interfere with existing compiler optimizations.

**Runtime libraries.** Libraries of parallel routines offer the advantage of language independence. Since they are not tied to any particular compiler, library routines can eliminate the need to rewrite or recompile programs when a system is modified. The library is simply replaced by a new version. Several versions of Fortran and C for parallel machines rely on high-level interfaces to libraries. The use of runtime routines is awkward and error-prone, however.[3,4,10] Parameter lists must be bulky to compensate for the fact that library units execute in isolation from the general program context.

Debugging is difficult since there is no clearly defined relationship with program structure or semantics. Another disadvantage is that, although the library approach appears to provide an easy means of integrating portability (by just recoding the library routines and leaving the invoking programs alone), this is not necessarily true. Parallel libraries often have such close ties to system architecture that porting programs to other machines results in inefficient or unreliable performance.

**What scientists choose.** Of the three alternatives for language support discussed here, scientific programmers rely on language extensions or runtime libraries. There are several reasons for this. First is the availability of production-level compilers for parallel machines. Manufacturers typically support Fortran (and recently, C) for scientific programming, but not the newer concurrent programming languages. The cost of developing translators makes extension of a previously supported language more attractive. Compiler availability still remains an issue when parallelism is supported by a language-independent library, since a compiler is needed to generate the code invoking the routines.

Familiarity is a second consideration. For practical reasons, an applications programmer is more likely to continue using a language than to learn a new one, even when it offers more expressiveness or flexibility. The continuing popularity of Fortran for large-scale numerical applications is due to the fact that most scientists and engineers learned to program in this language. The use of language extensions or libraries reduces the number of new constructs that must be learned and applied.

Yet another factor is the apparent ease with which sequential programs may be parallelized using extensions or libraries. What could be more straightforward than to add parallelism by inserting special statements or subroutine calls? Although this may not, in fact, be an easy or effective way to develop parallel programs, users find it practical. They are confident that inserting the "right" statements will produce the same results as the original program, and at a much faster rate.

There are other reasons that concurrent languages have not enjoyed much success among scientific users. Most were designed in response to the needs of the computer science community (that is, exploiting concurrency). Features for large-scale numerical calculations are underdeveloped. Even when numerical capabilities are adequate, the scientific user is faced with the need to learn a new approach to programming, since these languages represent radical departures from familiar structures. Switching languages also requires the recoding of substantial numbers of existing applications and limits the sharing of code with colleagues.

It can be argued that programmers should change in order to employ parallelism effectively, but the fact remains that they are reluctant to do so. Few users are willing to make this kind of investment without hard evidence that the new language will provide significant benefits for their types of applications on their particular machines.

This user predilection for extensions and libraries compromises the structural integrity of parallel programs. Instead of designing and implementing the program with parallel behavior in mind, parallelism is added after the fact in an *ad hoc* fashion. Not only does the process add yet another restructuring to program development, it invites other problems as well. Since the programmer believes that the sequential program was correct, testing of the parallel version may not be thorough. In addition, the current state of commercial language support — vendor-specific extensions or libraries — means that even the most portable sequential code will become machine-dependent in its parallel form.

## Parallelizing scientific code

In Figures 7 and 8, portions of the satellite example are parallelized. The language is IBM's Parallel Fortran (PF), but the number and type of statements required are similar to other parallel extensions of Fortran.[5,10] The code in Figure 7 shows the use of a parallel DO loop to apply a single set of operations concurrently to different data. This corresponds to the plane-wise transformation of array *A* during the backward sweep.

Parallel loops are the most common form of parallelization in Fortran programs. The structure resembles a normal DO loop, with the loop body subdivided to reflect the fact that multiple processors will be involved. The PF compiler automatically determines how to assign iterations to processors. The programmer, however, indicates which statements should be performed for every iteration (DO EVERY portion of the loop body) and which should be executed only once per processor (DO FIRST and DO FINAL sections).

The structure looks simple enough, but the partitioning of data among iterations can be tricky. The programmer must first identify any variable used to accumulate values within the loop. Each of these must be replicated for the multiple processors by declaring a new variable that is local to the loop body (such as the PRIVATE variable LTRANS, used for the accumulation of a partial sum corresponding to the number of transformations required).

Since Fortran provides no automatic initialization of local variables, this must be performed explicitly by the processor owning the copy (such as in the DO FIRST section). The situation is somewhat confused by the fact that most parallel Fortrans assume that the index variable is implicitly local and do not allow its declaration within the loop. The user must also ensure that any variables that will be updated by more than one processor are protected by a lock (in the example, the LOCK on DO FINAL ensures that ITRANS is accessed by only one processor at a time).

The nondeterminism inherent in parallel computing means that the results of a parallel loop are uncertain when the values generated by one iteration depend — directly or indirectly — on those produced by another iteration. Several current compilers can detect data dependences and inhibit parallelization, but the user is responsible for devising a solution. It is at this point that most scientific programmers start running into problems. Parallel Fortrans offer few alternatives for managing data. The programmer may be forced to make redundant copies of data, use multiple levels of indirection, or invent some other way to outwit the compiler's safeguards.

Figure 8 illustrates the code required to perform distinct sets of operations in parallel. This corresponds to the initial portion of the backward sweep, when four independent matrix computations are needed. Individual processes, called tasks in PF, must be created to perform the work. Note that the compiler no longer assumes responsibility for creating task units and assigning them to processors. Instead, the user must explicitly initialize tasks, map them to the available physical processors (the NPROCS intrinsic function determines how many processors are currently available), and assign work to them. The tasks must also be terminated explicitly. Since task operations are costly, the programmer may find it more efficient to "save" tasks needed later in the program. In our exam-

```
C  Number of transforming operations will be counter
         ITRANS = 0
C Loop structure will make use of as many processors as possible,
C up to the number of iterations (i.e., columns in array A)
C Each iteration transforms one plane
         PARALLEL LOOP 810 I = 1,DIMEN
C Declare variables that must be private to each processor
C (the index I is private by default)
         PRIVATE (LTRANS)
                  ...
C Initialization for the iterations assigned to each processor
         DO FIRST
              LTRANS = 0
                  ...
C Independent loop body code to transform A(I) plane
         DO EVERY
              DO 820 J = I TO DIMEN
                  ...
                   LTRANS = LTRANS + 1
                  ...
  820    CONTINUE
C Add local sum to shared variable accumulating total transformations
         DO FINAL LOCK
              ITRANS = ITRANS + LTRANS
  810 CONTINUE
```

**Figure 7. Parallelized looping structure, satellite example.**

```
C Use four processors if available
       N = MAX(NPROCS(),4)
C Create and initialize tasks and save their IDs
C (they will be ready for scheduling work at the WAIT statement)
       DO 510 I = 1,N
            ORIGINATE ANY TASK IDTSKS(I)
            SCHEDULE TASK IDTSKS(I), CALLING INITIAL
  510 CONTINUE
C Perform in parallel: B1 x B2, invert C, eigenvalues for D and E
       WAIT FOR ANY TASK NXTTSK
       SCHEDULE TASK NXTTSK,
      *  CALLING MATMULT(B1,B2,RESLT1)
       WAIT FOR ANY TASK NXTTSK
       SCHEDULE TASK NXTTSK,
      *  CALLING INVERT(C,CINV)
       WAIT FOR ANY TASK NXTTSK
       SCHEDULE TASK NXTTSK,
      *  CALLING EIGEN(D,DIMEN,MAXIT,TOL,TCONSTR)
       WAIT FOR ANY TASK NXTTSK
       SCHEDULE TASK NXTTSK,
      *  CALLING EIGEN(E,DIMEN,MAXIT,TOL,TCONSTR)
C Delay until all computations are complete
       WAIT FOR ALL TASKS
C Terminate the tasks
       DO 520 I = 1,N
            TERMINATE TASK IDTSKS(I)
  520 CONTINUE
```

**Figure 8. Parallelized subroutine invocation, satellite example.**

ple, the tasks would be initialized at the start of the program and continued until all successor sweeps are completed, rather than being recreated at the start of each sweep.

The parallelizing code in Figure 8 consists primarily of new statements, but the functions to multiply and invert matrices also had to be converted to subroutines to fit the task invocation framework. More radical alterations are required to share data among tasks. In general, variables are local to each task unless they are passed as parameters or occur in global storage (called COMMON in Fortran).

The programmer must include a lock or equivalent mechanism to ensure that ordering is preserved when a global variable is referenced or altered by a task. These concepts appear to relate well to data-storage features in sequential Fortran. Consequently, many scientific programmers assume that a program may be safely parallelized by adding a lock to each shared variable. This is not usually true. The precise meaning of local or global storage is affected by a number of subtleties related to the memory model of the parallel machine. It may be necessary for the programmer to distinguish variables that are local to a subroutine invocation from those that are local to a processor (which may run two or more of the parallel subroutines).

Explicit copy operations may be required to initialize these variables. Other statements may be needed to update COMMON blocks explicitly; PF, for example, requires that the user indicate whether COMMON values should be copied at the beginning of a parallel task, at the end, or both. These notions are counter-intuitive to scientific programmers, whose experience with scoping and storage management has been limited to the simple model of sequential Fortran.

It is important to note that the structures provided by PF are not at a consistent level of abstraction. The parallel DO is high-level, with implicit creation/termination of tasks and load balancing among processors; in some cases, the compiler may even extend the lifetime of tasks between looping structures to improve performance. The subdivisions of the loop body provide implicit mechanisms for restricting the number of times an operation is performed. An implicit barrier is also created at the end of the loop so that the processors wait until all iterations are complete before continuing.

The parallel invocation of subroutines, on the other hand, is quite low-level. The programmer must explicitly control all task operations and processor mapping. Since there are no implicit timing or sequencing mechanisms, any intertask coordination must be accomplished using low-level constructs based on events (synchronizing signals sent from one processor to another).

Facilities for controlling access to data are also provided at contradictory levels. In the parallel DO construct, the LOCK option uses an implicit lock to transform DO FIRST or DO FINAL code into the equivalent of a critical section (a sequence of instructions that can be executed by only one processor at a time). For parallel subroutines, the user must explicitly create and destroy a series of named locks. The operations provided for managing locks are so primitive that a lock's identifier must be passed as an argument if it is to be shared by multiple tasks.

Although these examples are specific to PF, similar inconsistencies exist among other parallel language extensions. Even the parallel libraries are guilty of mixing high- and low-level structures at random (see the Sequent Balance examples in reference 3).

Restrictions on the nesting of parallel constructs may require the reformulation of sequential control structures as well. To keep from nesting a parallel loop inside a sequential one, for example, the programmer may have to "unroll" the outer loop, replacing it with multiple copies of the statements forming the loop body. Other restrictions may require that sequential subroutines or functions be substituted in-line (replaced by a copy of the code in which references to parameters have been replaced by the corresponding actual arguments). Restructurings such as these introduce many new possibilities for error as well as affect readability.

## The plight of the programmer

All in all, scientific programmers are unlikely to find language structures that map cleanly to their parallelization needs. This is disturbing because these users would benefit most from the error detection and comprehensibility provided by appropriate high-level features. Instead, parallel compilers can find and report only the most blatant errors. Programmers are forced to juggle inadequately described and potentially dangerous operations. In many cases, the compiler obeys directions even when the code is likely to produce incorrect results. This is effectively a leap backward in time: as in the early days of Fortran, parallel features are closer to symbolic assembly code than to high-level constructs.

In addition, the primitives used to specify parallelism are closely tied to the underlying machine. It has been demonstrated that management of the architectural configuration determines to a great extent the efficiency, effectiveness, and reliability of parallel implementations.[8,9] Unfortunately, it is equally clear that full responsibility lies with the programmer, who must now be concerned with the optimal scheduling and binding of processes to processors and the distribution of data to memory locations.[3,11]

Unlike the sequential programming environment — where scientists learned to develop applications — parallel systems lack the buffering effect of logical independence. Parallelism should be incorporated at a reasonable level of abstraction rather than simply providing a notationally convenient way of specifying what are, in fact, machine-specific operations.

Although we look forward to the day when real-world problems may be mapped to parallel hardware seamlessly and automatically, the fact remains that parallelism is still in the embryonic stage. Little is known about effective techniques for conceptualizing and formalizing parallel strategies. It's not surprising that scientific programmers entertain a number of misconceptions about parallel machines and programs. In spite of their criticisms of "von Neumann programming," a large majority of the user community still views computation as a sequence of operations that transforms data. Parallel processing is correspondingly visualized as the simultaneous execution of those sequences.

This "extended sequential" view of parallelism is misleading. It implies that a programmer need only partition the sequential solution to achieve reliable and repeatable results. This is not the case.[12] The effects of nondeterminism on parallel behavior are still being explored, and we do not yet know how to harness this property reliably. According to McGraw and Axelrod, "The fact that a [parallel] program functions correctly once, or even one hundred times, with some particular set of inputs, is no guarantee that it will not fail tomorrow with the same inputs."[9]

Misconceptions about the nature of parallelism also lead to unrealistic performance expectations. Many users are confused when parallelization does not achieve a speedup proportional to the number of

**Table 1. Desirable characteristics of parallel languages — two viewpoints.**

| Category | For Scientific Researcher | For Computer Scientist |
|---|---|---|
| Convenience | Fortran 77 syntax | Structured syntax and abstract data types |
| | Minimal number of new constructs to learn | Extensible constructs |
| | Structures that provide low-overhead parallelism | Less need for fine-grain parallelism |
| Reliability | Minimal number of changes to familiar constructs | Changes that provide clarification |
| | No conflict with Fortran models of data storage and use | Support for nested scoping and packages |
| | Provision of deterministic high-level constructs (like critical sections, barriers) | Provision of nondeterministic high-level constructs (like parallel sections, subroutine invocations) |
| | Syntax that clearly distinguishes parallel from serial constructs | Syntactic distinctions less critical |
| Expressiveness | Conceptual models that support common scientific programming strategies | Conceptual models adaptable to wide range of programming strategies |
| | High-level features for distributing data across processors | High-level features for distributing work across processors |
| | High-level control over locality of data accesses | High-level control over locality of work |
| | Parallel operators for array/vector operands | Parallel operators for abstract data types |
| | Operators for regular patterns of process interaction | Operators for irregular patterns of process interaction |
| Compatibility | Portability across range of vendors, product lines | Vendor specificity or portability to related machine models |
| | Conversion/upgrading of existing Fortran code | Conversion less important (formal maintenance procedures available) |
| | Reasonable efficiency on most machine models | Tailorability to a variety of machine models |
| | Interfacing with visualization support routines | Minimal visualization support |
| | Compatibility with parallel subroutine libraries | Little need for "canned" routines |

processors. Their failure to comprehend the impact of architecture on program behavior leads to problems with reusability and portability as well.

Take, for example, the effects of the machine's memory model. Most scientific programmers equate a shared memory model with the single address space of uniprocessing (that is, all processors have equal access to all data values at all times). This leads to the correct assumption that explicit mechanisms must be used to protect global data that will be modified. It also leads to a fallacy responsible for many program errors. Many shared-memory systems do not actually provide a single, homogeneous address space. Their hierarchical distribution of data into levels (such as local cache, shared cache, and extended memory) means, at best, that some time may elapse between the updating of a value and its propagation throughout the system. At worst, the programmer may be held responsible for providing explicit mechanisms to ensure that the values are propagated correctly.

These programming challenges are similar to those faced by scientific users when they converted to vector systems. It is important to note that it has taken a dozen years for programmers to gain proficiency with vectorizing compilers. As Hack comments, "... the constraints have become much more complex, while the penalty for inefficient utilization of the system is substantially larger."[11] Users need language structures that encourage a realistic approach to developing and testing parallel applications.

If our arguments sound suspiciously like the justifications for high-level programming languages and structured programming methodologies we heard two decades ago, that is because parallel programming today faces similar problems. In reflecting a low-level view of concurrent execution that reinforces user misconceptions, parallel languages do the scientific community a disservice. This approach not only increases the frustration and expense of program development, but also raises questions about the reliability of program results. Furthermore, it is short-sighted to tie program development so closely to specific systems. By implementing parallelism in machine- and translator-specific fashion, we limit the program's reusability as well as its applicability to the improved systems that lie ahead.

The present level of language support for parallel programming requires that the user expend more effort in managing the problem-solving resource than in actually solving the problem. This may be a positive factor for computer scientists. As Hudak points out, "We as programmers usually have more to say about a problem than just the answer. We typically have a specific data representation in mind and we might know a better way to run the program on a particular architecture."[13]

The key here is that while computing professionals should be able to apply configuration-specific expertise, it is counterproductive to expect the same of the general user community. Support for scientific applications is inadequate as long as parallelism must be expressed in terms of a particular machine or memory model. Scientists turn to parallel processing in an attempt to understand the physical world through modeling and simulation. They need language features that improve the comprehensibility and accuracy of parallel programs.

Table 1 summarizes the characteristics

desirable in parallel languages, contrasting the viewpoints of the scientific researcher with those of the computer scientist. A few features are needed by all parallel programmers:

- facilities for performing I/O operations over parallel channels;
- simple, orthogonal, and high-level contructs; and
- compiler detection of common parallel errors.

In many ways, however, the needs of the two user communities are distinct.

## New directions

The problems facing scientific programmers are being attacked from several perspectives. Jack Dongarra and Dan Sorenson of Argonne National Laboratories have developed a tool kit to increase the portability of parallel programs.

The Schedule package provides a standardized interface between the user and the parallel Fortrans provided by major vendors. The programmer begins with a sequential Fortran program and identifies any code that can execute in parallel by encapsulating it in subroutines. A "data dependency graph" is constructed to indicate how many replications of which subroutines can run in parallel and the order in which they must be executed. The code is then modified by replacing the subroutine calls with invocations of Schedule routines, which provide a system-independent way to manage task creation, assign work to processors, etc.

Although the concept is simple, programming with the tool kit can be extremely complicated. Like parallel runtime libraries, Schedule requires lengthy parameter lists that bear little resemblance to normal Fortran code. Another problem is that the package adds an extra layer between the user and the machine, which makes error detection almost impossible. The toolkit's principal advantage is that it is readily available as public domain software through supercomputing centers and national labs.

Dongarra and Sorenson also motivated another project designed to increase the portability of parallel scientific applications. The developers of Linpack and Eispack — popular libraries of routines for linear algebra and eigen systems — have joined forces with the Numerical Algorithms Group from Oxford University to create a new library exploiting recent developments in parallel algorithm design. Their proposed product, Lapack, will make efficient implementations of standardized numerical techniques available on most parallel systems.

A consortium called the Parallel Computing Forum has proposed a different strategy to counteract portability problems. The PCF includes all major supercomputer manufacturers, with observers from national supercomputing facilities. Its primary effort has been the drafting of a new standard definition for parallel Fortran named PCF-Fortran. The goal is to provide parallel extensions to Fortran 77 that allow scientific code to run quickly and efficiently on a variety of machines. Since all corporate members of the PCF will implement the language, parallel programs will be portable across a wide range of architectures.

The small size and clearly delineated goals of the PCF have contributed to a spirit of cooperation among its members. The PCF-Fortran draft standard was first released for public review and comment in August 1986 (four major revisions have been issued since that time, the most recent dated June 1990). The American National Standards Institute subcommittee X3H5 will handle the standardization process. The new draft is scheduled for initial release in January 1991.

Language developers are also trying to facilitate parallelization by providing parallel features at higher levels of abstraction. The PCF-Fortran standard extends the functionality of parallel loops with features that allow the programmer to indicate that execution of successor iterations must delay long enough to allow a predecessor to generate some needed value. This process takes care of many data-dependency problems that plague current implementations.

Other interesting high-level constructs include critical sections, single process sections (code embedded in a parallel construct that will be executed by just one processor) and parallel regions (designed to provide low-overhead parallelism). Unfortunately, PCF-Fortran continues the tradition of combining high- and low-level features in a random mixture.

Supercomputer Systems, Inc. — a company formed by Steven Chen and other members of the Cray X-MP design team — has taken a novel approach to parallel language design. Recognizing the need for interaction between language developers and target users, SSI initiated a collaborative effort with Cornell University. SSI experts have devised a minimal set of high-level language constructs needed to parallelize existing scientific code. Cornell staff evaluate the usefulness of the constructs in large-scale applications carried out by scientific and engineering researchers on the Cornell National Supercomputer Facility. The insights gained through actual experience are relayed back to SSI in the form of recommendations and user commentary.

One shortcoming of current parallel languages is their concentration on features for dividing the work to be performed in parallel; no high-level provisions are made for distributing data. The proposed languages that we describe perpetuate this problem. New features were designed to facilitate the control of program operations, not to provide mechanisms for storing and managing large quantities of data for access by multiple processors.

The Linda system developed at Yale University offers a revolutionary approach to the problems of data sharing in parallel programs. Linda is not a programming language but a collection of operations added to a sequential language to create a parallel extension (like C-Linda, Fortran-Linda, or Scheme-Linda).

Linda replaces the traditional concept of program storage with an abstract model of "tuple space."[7] When a process begins to communicate or share data with another process, it adds a new object to the tuple space. The second process can then access the information. The programmer does not need to know anything about how the parallel machine stores or shares data. Everything occurs transparently through the Linda primitives.

Although Linda was originally used only within the academic community, it now appears to have gained the support of computer manufacturers. Cogent, for example, recently announced development of C++ and Fortran versions to be compatible with (and portable to) such competitive products as Sequent's Balance and Symmetry, Encore's Multimax, and Intel's iPSC computers. This is a major step forward, since Linda's goal is to provide flexibility and expressiveness for a variety of programming paradigms through machine-independent constructs.

Recent efforts in parallel debugging tools also may have significant impact on scientific programming. Currently available parallel debuggers generate extreme quantities of low-level data. As the technical difficulties confronting designers are resolved, however, more attention is being given to the question of how information should be presented to the user.[12,14] New techniques are emerging for abstracting the performance data obtained through static and dynamic analysis to provide a high-level, graphical view of parallel program behavior.

All of these developments offer promise for the future of parallel computing. More important, they are evidence that a concerted effort is being made to build on constructs already familiar to the scientific user community. This approach should benefit both sides of the community. Developers will find wider acceptance of their products, while users will enjoy a shorter learning curve.

Given the general lack of experience in parallel software, it is not surprising that scientific programmers are finding it difficult to parallelize their applications. Programming languages constitute the primary point of contact between scientist and machine. Every effort must be made to ensure that these languages benefit from the proven strengths of the sequential approach: (1) maintaining logical independence to shield the user from unnecessary physical details; and (2) providing a clear, consistent relationship between language structures and the descriptive needs of the user.

The mechanics of program development provide the added insight that extensions to familiar sequential languages — through parallel constructs or high-level interfaces to libraries — are more likely to appeal to scientific programmers than are new concurrent languages.

It is our responsibility as computer scientists to look beyond our own experiences in parallel programming. If parallelism is to realize its full potential, we must understand the needs and expectations of the user community and devise new ways to facilitate the development of parallel applications. As Bailey recently observed, "We need to work harder at making supercomputers not just connected to remote locations but accessible to do science. Researchers must be free to concentrate on their research, not struggle with [machine-dependent] quirks and minute details."[15]

There is still a sizable gap between the user's conceptual solution to a problem and its ultimate realization on a parallel machine. Only by narrowing that gap can we increase the reliability and effectiveness of scientific parallel programming. ■

## Acknowledgments

A portion of this research was conducted at the Cornell National Supercomputer Facility, a resource of the Center for Theory and Simulation in Science and Engineering (Cornell Theory Center). The center receives major funding from the National Science Foundation and IBM as well as support from New York State and members of the Corporate Research Institute. Additional funding was provided through Auburn University Research Grant-in-Aid 88-217.

## References

1. D. Bergmark, *History of Parallel Processing on the CNSF*, tech. report, Cornell Theory Center, Cornell University, Ithaca, N.Y., to be published in Dec. 1990.

2. G. Bell, "The Future of High Performance Computers in Science and Engineering," *Comm. ACM*, Vol. 32, No. 9, Sept. 1989, pp. 1,091-1,101.

3. M. Kallstrom and S.S. Thakkar, "Programming Three Parallel Computers," *IEEE Software*, Vol. 5, No. 1, Jan. 1988, pp. 11-22.

4. S. Ranka, Y. Won, and S. Sahni, "Programming a Hypercube Multicomputer," *IEEE Software*, Vol. 5, No. 5, Sept. 1988, pp. 69-77.

5. R.G. Babb II, ed., *Programming Parallel Processors*, Addison-Wesley, Reading, Mass., 1988.

6. N. Wirth, "On the Design of Programming Languages," *Proc. IFIP Congress 74*, North-Holland, New York, 1974, p. 387.

7. N. Carriero and D. Gelernter, "Linda in Context," *Comm. ACM*, Vol. 32, No. 4, Apr. 1989, pp. 444-458.

8. A. Karp, "Programming for Parallelism," *Computer*, Vol. 20, No. 5, May 1987, pp. 43-57.

9. J. R. McGraw and T.S. Axelrod, "Exploiting Multiprocessors: Issues and Options," *IEEE Software*, Vol. 5, No. 5, Sept. 1988, pp. 7-25.

10. A.H. Karp and R.G. Babb II, "A Comparison of 12 Parallel Fortran Dialects," *IEEE Software*, Vol. 5, No. 5, Sept. 1988, pp. 52-66.

11. J.J. Hack, "On the Promise of General-Purpose Parallel Computing," *Parallel Computing*, Vol. 10, No. 3, 1989, p. 273.

12. C.E. McDowell and D.P. Helmbold, "Debugging Concurrent Programs," *ACM Computing Surveys*, Vol. 21, No. 4, Dec. 1989, pp. 593-622.

13. P. Hudak, "Exploring Parafunctional Programming: Separating the What from the How," *IEEE Software*, Vol. 5, No. 1, Jan. 1988, p. 58.

14. S. Utter and C.M. Pancake, "Advances in Parallel Debuggers: New Approaches to Visualization," *Advances in Parallel and Distributed Computing*, to appear in the inaugural issue, Jan. 1991.

15. F.R. Bailey, "Toward an Improved Supercomputing Environment," *Supercomputing Review*, Vol. 1, No. 1, June 1988, p. 18.

**Cherri M. Pancake** is an assistant professor of computer science and engineering at Auburn University and a visiting scientist at the Center for Theory and Simulation in Science and Engineering, Cornell University. Her research focuses on parallel programming tools for performance visualization, parallel debugging, and interactive parallelization.

Pancake received a BS degree in design and environmental analysis from Cornell University and a PhD degree in computer science and engineering from Auburn University. She is a member of the IEEE, IEEE Computer Society, Phi Kappa Phi, Upsilon Pi Epsilon, Phi Beta Delta, and a chapter officer of ACM.

**Donna Bergmark** is a senior manager at the National Supercomputer Facility at Cornell University. Her research interests are in the areas of parallel processing and scientific computing, with emphasis on compilers for parallel processors.

Bergmark received an MS degree in computer science from Cornell University. She has been a member of the ACM since 1969.

Readers may contact Pancake at Auburn University, Department of Computer Science and Engineering, Auburn, AL 36849.

EDAC

# The European Design Automation Conference

## Amsterdam, The Netherlands, 25-28 February 1991

EDAC-91 will be held in Amsterdam. The Programme Committee accepted 101 verbal presentations organized in three parallel sessions. Two poster sessions are planned as well.

The scope covers all areas of the design process, from concept to manufacture, and includes CAD and DA tools for analog, digital, VLSI, microwave and high-speed electronics.

In addition, three panel sessions, a number of fringe meetings, a day of tutorials and vendor presence are organized.

EDAC Conferences are run by EDAC, a non-profit association, in cooperation with CAVE (CEC), ESPRIT Basic Research Action (CEC DG 13), IBM Nederland N.V., IEEE Computer Society DATC, IFIP Working Group 10.2, IFIP Working Group 10.5 (pending), PTT Research Nederland, and SCME (Stichting Centra voor Microelektronica).

For information, contact:

EDAC-91 Secretariat
CEP Consultants Ltd.
26-28 Albany Street
Edinburgh EH1 3QH
Scotland
Tel.: +44 31 557 2478
Fax: +44 31 557 5749

**General Chair: Jochen JESS**
**Programme Chair: Hugo de MAN**
**Past Chair: Gordon ADSHEAD**

IEEE COMPUTER SOCIETY    THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

# Compiling Scientific Code Using Partial Evaluation

Andrew Berlin, Massachusetts Institute of Technology

Daniel Weise, Stanford University

Scientists are faced with a dilemma: They can write abstract programs that express their understanding of a problem but do not execute efficiently, or they can write programs that execute efficiently but are difficult to write and understand. Partial evaluation can solve this problem by providing the missing link between the code presented to the compiler and the computation envisioned by the programmer.

**Partial evaluation transforms a high-level program into a low-level program that is specialized for a particular application. This exposes the parallelism inherent in the underlying numerical computation.**

Partial evaluation is a technique for converting a high-level program into a low-level program specialized for an application. Rather than just considering a program's code, the compiler can also consider information available at compile time about the data structures the program will manipulate. Scientific applications often provide enough information at compile time to allow advance data manipulation, leaving only the underlying numerical computation to be performed at runtime. This approach eliminates nearly all of the programmer's data abstractions and control abstractions at compile time, producing high-performance code.

We have implemented a prototype compiler that uses partial evaluation. Experiments with our compiler have shown that for an important class of numerical programs, partial evaluation can provide dramatic performance improvements: We have measured speedups over conventionally compiled code that range from seven times faster to 91 times faster. These experiments have also shown that by eliminating inherently sequential data-structure references and their associated conditional branches, partial evaluation exposes the low-level parallelism inherent in a computation. By coupling partial evaluation with parallel scheduling techniques, this parallelism can be exploited on heavily pipelined or parallel architectures. We have demonstrated this approach by applying a parallel scheduler to a partially evaluated program that simulates the motion of a nine-body solar system.

## Abstraction and high-level programs

High-level languages such as Lisp are very powerful in that they allow computations to be expressed in terms of abstract numerical methods and techniques, using abstractions to mirror the way a person thinks about a problem. This is in contrast to the programming methodology associated with mid-level languages such as Fortran, in which programmers apply the numerical techniques themselves to derive the numerical computation required for a particular problem, and then use the programming language only to express the results of their efforts.[1]

Programs can be classified according to their versatility and ease of construction. At the lowest level are programs that can be applied to only one problem, such as the

```
(define (make-integrator F time-step) ;;;make-integrator takes as arguments
                                      ;;; the function to be integrated, F,
                                      ;;; and the time-step.
  (define (produce-next-state current-state)
    (define k0  (scale-system time-step (F current-state)))
    (define k1  (scale-system time-step
                 (F (add-systems current-state (scale-system 1/2 k0)))))
    (define k2  (scale-system h
                 (F (add-systems current-state (scale-system 1/2 k1)))))
    (define k3  (scale-system h
                 (F (add-systems current-state (scale-system 1/2 k3)))))
    (define new-state
      (scale-system 1/6
        (add-systems k0
          (scale-system 2 k1)
          (scale-system 2 k2)
          k3)))
    new-state) ;;produce-next-state returns new-state
  produce-next-state) ;;;make-integrator returns produce-next-state
```

**Figure 1. Fourth-order Runge-Kutta integrator. This high-level program composes existing functions to literally construct a new procedure that performs an integration step.**

## Overview of Scheme

Scheme is in the Lisp family of languages. All objects, whether they are data structures or continuations, are dynamically created and have indefinite extent. This means that they can be created at any point. Once created, they are reclaimed by the storage system only when a program drops all references to them. The primitive data types in Scheme include numbers, lists, vectors, and procedures.

The different types of Scheme expressions used in this article are as follows:

| | |
|---|---|
| (define <name> <exp>) | <name> is defined to have the value returned by <exp>. |
| (define (<name> <f1> ... <fn>) <exp>) | This expression defines a procedure having name <name>, formal parameters <f1> through <fn>, and body <exp>. |
| (let <binding-list> <exp>) | <binding-list> is a list of name-expression pairs. The expressions are evaluated. The resulting values are bound to the names, and then <exp> is evaluated. |
| (let* <binding-list> <exp>) | This expression is like let, except that the bindings are processed serially: Each name-expression pair is evaluated and bound in turn. |
| (if <pred> <then> <else>) | First <pred> is evaluated. If it evaluates to true, the <then> expression is evaluated; otherwise the <else> expression is evaluated. |
| (<exp1> <exp2> ... <exp3>) | When the first element of an expression is not a reserved keyword such as if or define, an expression denotes a function call. Each expression is evaluated, and then the result of evaluating the first expression is applied to the other values. |

Here are some of Scheme's built-in functions:

| | |
|---|---|
| +, -, *, / | Perform arithmetic operations. |
| vector | Creates a one-dimensional array. |
| vector-ref | Retrieves an element from a one-dimensional array. |
| vector-length | Computes the length of a one-dimensional array. |
| cons | Creates a pair (a tuple of length two). |
| car, cdr | Retrieve the first and second element of a pair, respectively. |

three-body problem, the analysis of a given dam under different loads, or the transient behavior of a particular circuit for different inputs. These programs, because they solve only one problem, are very efficient. Unfortunately, they are rarely worth writing by hand, since their usefulness is limited to one particular problem.

At the middle or conventional level are programs typically written in C or Fortran that solve a class of problems, such as programs for solving the *n*-body problem, analyzing dams, or simulating circuits. They are more versatile than the lowest class, but less efficient.

At the highest level are programs constructed using such advanced abstractions as higher order procedures, automatic storage mechanisms, and object-oriented methods. These programs, usually written in Lisp or Smalltalk, embed representation and control choices in the data objects being manipulated. They are the easiest to construct and the most versatile, because they can be adapted and reused more readily than conventional programs. But they are the least efficient because of the computational cost imposed by the abstraction mechanisms.

As an example of a high-level program, consider the problem of numerical integration of an unknown function *F* that computes the rate at which a system changes. When given a function *F*, the program in Figure 1 dynamically creates a new procedure that performs the integration. (See sidebar at left for an overview of Scheme, the language used in this article.) Notice that this program is totally independent of the function being integrated, the data structures used to represent the system state, and the storage-allocation strategy. The procedure make-integrator takes as input the function to be integrated and the integration time step. It then creates and returns a new procedure. When run, this procedure takes as input the current system state and then performs an integration step to produce the system state corresponding to one time step later.

This style of programming is quite flexible. The code for the integrator can be used in many different applications, making feasible a very general library of numerical techniques that operate independently of data representations and storage-maintenance strategies. Roylance[1] and Halfant and Sussman[2] give more detailed and powerful examples of abstraction in numerical computation.

The same flexibility that makes high-level languages expressive also reduces their efficiency. High-level programs are inefficient because maintaining abstraction mechanisms — dynamic storage allocation, object-method dispatching, and higher order procedures — requires computation. Also, because of its general nature, an individual procedure does not provide enough information for the compiler to predict the computation needed. For example, efficiently compiling the make-integrator program shown in Figure 1 would be quite difficult — the compiler does not know what function will be integrated or what kind of data structure add-systems will manipulate.

Conventional compilation can improve high-level program performance by optimizing references to variables such as time-step, passing parameters in registers, placing small functions in line, and performing interprocedural analysis.[3] But the performance of compiled programs still falls far short of that of the low-level numerical programs an expert programmer would write: The high-level aspects of the program, such as the procedure calls and data-structure manipulations, remain in the compiled program, imposing a performance penalty. This inefficiency remains because static analysis considers only the code for a program — the instructions for manipulating the data — not information about the data itself.

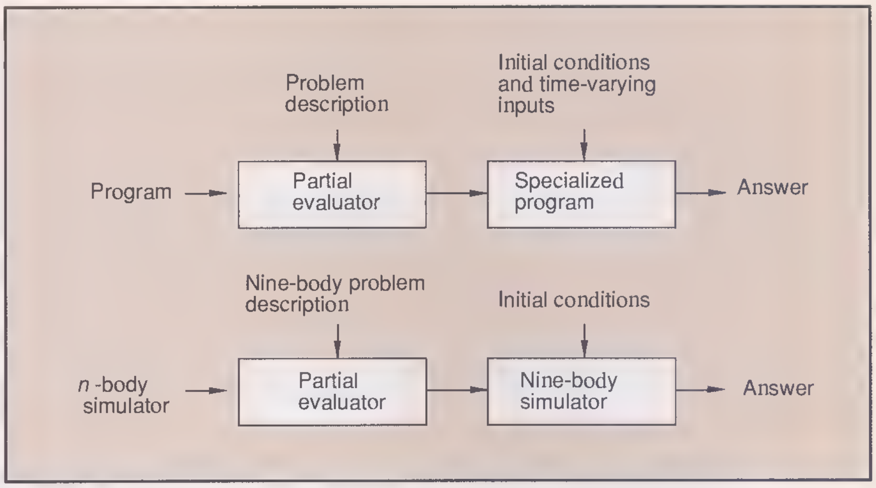


**Figure 2. A partial evaluator transforms a general program into one specialized for a given problem.**

## Compiling for a particular problem

Partial evaluation transforms a general (high or mid-level) program into a specialized (low-level) program by taking advantage of information available at compile time about the data structures the program will be run on. As Figure 2 shows, given a high-level program that computes the motion of a collection of planets and the fact that the particular problem being studied involves exactly nine planets, partial evaluation produces a low-level program that computes the motion of a nine-planet solar system for varying initial conditions.

This strategy differs from conventional compilation techniques. Conventional compilers seek to optimize the execution of procedure calls and data-structure manipulations, whereas partial evaluation seeks to eliminate these operations by performing them in advance, at compile time.

Partial evaluation is especially effective on scientific programs because these programs have a special property: They are mostly data independent. A program is data independent when the sequence of operations it performs does not depend on the results of the computation. For example, for any given matrix size, matrix-multiply is data independent: It performs a fixed set of multiplications, regardless of the numerical values of the numbers being multiplied.

Data independence makes it possible to predict what operations a program will perform, even before actual numerical values for its inputs are available. This allows data manipulation operations to be performed in advance — at compile time — leaving only the underlying numerical computation to be performed at runtime.

Many data-dependent programs become data independent once information is available about the problem that the program will be used to solve. For example, a general version of matrix-multiply, in which the size of the matrix is not known at compile time, would be data dependent, since the sequence of operations would vary depending on the size of the matrices being manipulated. This would prevent the matrix reference operations from being performed at compile time, requiring that the matrix data structures be manipulated at runtime. However, by considering information about the matrices associated with a given problem, the matrix size can be determined at compile time, transforming matrix-multiply into a data-independent program.

## Partial evaluation of data-independent programs

There is a very simple way to derive the underlying numerical computation expressed by a data-independent program: Simply execute the program at compile time and keep track of what it does. The key idea is to capture information about how a program solves a given problem. To do this, run the program on input data structures that correspond to the problem statement. Although the actual numerical values for some pieces of data will not be known until runtime, their location within the data structures will be known at compile time. Numerical values not yet available are represented symbolically using a data structure known as a placeholder. Placeholders can also hold additional information about a missing number, such as its type.

For example, consider the input data structures for a program that integrates the motion of the solar system. The program takes as input the current positions and velocities of the planets, and produces the positions and velocities corresponding to one time step later. Figure 3a shows typical input data at runtime. Because the planets are in different positions each time the program is run, numerical values for the positions are not known at compile time. Nonetheless, their locations in the data structures and their types are known, as expressed in Figure 3b.

```
Part A

 ;; Typical data at runtime:
 (define mars
    (make-planet 'mars
       (/ 1 3093500) ;mass
       (3-vector -1.295477589 -.8414136141 -.3513513446) ;position
       (3-vector .3440042605 -.3696674843 -.1789373952))) ;velocity

Part B

 ;; Data structure describing a specific problem:
 (define mars
    (make-planet 'mars
       (/ 1 3093500) ;The mass of a planet is known at compile time.
       (3-vector  (MAKE-PLACEHOLDER 'mars-position-x 'floating-point) ;p
                  (MAKE-PLACEHOLDER 'mars-position-y 'floating-point)
                  (MAKE-PLACEHOLDER 'mars-position-z 'floating-point))
       (3-vector  (MAKE-PLACEHOLDER 'mars-velocity-x 'floating-point) ;v
                  (MAKE-PLACEHOLDER 'mars-velocity-y 'floating-point)
                  (MAKE-PLACEHOLDER 'mars-velocity-z 'floating-point))))
```

**Figure 3. Data structure for a program that integrates solar system motion: (a) with typical runtime data, (b) at compile time, with placeholders.**

When the program is executed at compile time, placeholders are treated just like numbers. For example, they can be aggregated to form lists, stored in variables or vectors, and passed as arguments to procedures. Anything that manipulates a number will also manipulate a placeholder. This allows all data-manipulation operations (for example, procedure calls and data-structure manipulations) to be performed at compile time.

Our implementation of partial evaluation produces two values: a list of instructions and a result value, which is usually a placeholder or a data structure containing placeholders. We built the partial evaluator on top of a Scheme interpreter by modifying the behavior of its lowest level numerical operations.

During partial evaluation a numerical operation that encounters numeric arguments proceeds normally, returning a numeric result. A numerical operation that encounters placeholders returns a new placeholder as output and delays itself until runtime by appending an instruction to the list of instructions (see Table 1). The compiler combines the results of partial evaluation into a specialized program. The sidebar at right shows how partial evaluation works with an inner-product program.

## Data-dependent programs

Partially evaluating a program via symbolic execution works well for data-independent computations but runs into problems when applied to data-dependent computations. Most programs contain conditional branches, such as the If statement, in which a predicate is evaluated to determine whether to execute the code associated with the consequent or with the alternative. For data-independent computations, the predicate can always be evaluated at compile time, since it never depends on the data being manipulated. However, in data-dependent computations, the predicate can depend on values not computed until runtime.

Certain types of data-dependent conditionals can be partially evaluated by executing (hence generating code for) both the consequent and the alternative of the conditional branch at compile time. A conditional branch is then inserted into the compiled program to choose at runtime which set of code to execute. This approach is adequate for simple selection operations, such as those associated with the absolute value, Min, and Max functions, but it breaks down when used on recursive functions. Our compiler requires the programmer to declare explicitly (via a program annotation) the data-dependent conditionals that can be expanded in this fashion.

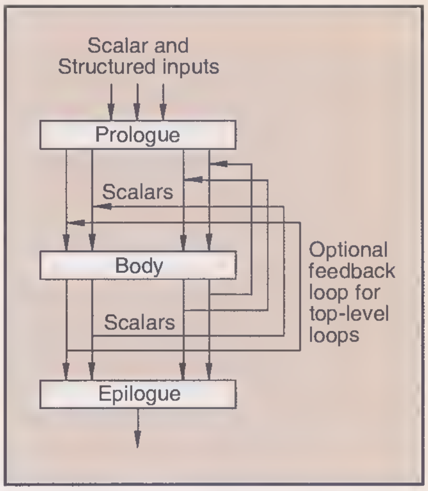
There are several ways to get around the problems associated with data dependencies. The simplest method, and the one we take, is to divide the program into data-independent regions, each of which can be partially evaluated. Such division limits the scope of the partial evaluation optimizations, since the data structures that act as interfaces between the data-independent regions of the program cannot be eliminated. Fortunately, with scientific codes, a programmer can make the data-independent regions of a program extremely large (often several thousand operations) by considering information about the problem that a program will solve. The data-dependent conditionals then occur only at the ends of these long computations for such operations as convergence checks and strategy selection.

## The prototype compiler

We have implemented a prototype compiler that uses partial evaluation. This compiler generates compiled code in a generic register-transfer language, in C syntax, or in Scheme syntax. It provides support for invoking partially evaluated programs as subroutines, in the same manner as the original Scheme programs from which they were derived. Since the original Scheme programs use high-level data structures to receive their inputs and return their results, the compiler generates a set of interface routines to convert between the scalar numerical values manipulated by the partially evaluated subroutine and the data structures used in the calling program.

The programs produced by our compiler have three stages: a prologue, a body, and an epilogue (see Figure 4). Partially evaluating a program yields two values: a result value *V*, which may be a placeholder or a data structure containing placeholders, and a list of instructions to be executed at runtime, which we call the body. The body takes as input numerical values for each input placeholder and performs the remaining numerical calculations that could not be performed during partial evaluation because of missing data. The prologue destructures the input data structures presented to the program at runtime, extracting a numerical value for each input placeholder. Similarly, the epilogue constructs the data structures that the program is expected to return, based on the values of the result placeholders. When the body is the body of a loop, the body can loop back directly to itself, which saves creating an output data structure and then destructuring it.

Our compiler automatically generates the prologue's destructuring instructions using



Scalar and Structured inputs
Prologue
Scalars
Body
Scalars
Epilogue
Optional feedback loop for top-level loops

**Figure 4. The three stages of programs produced by the compiler.**

**Table 1. Partially evaluating the expression (let ((*x* (+ *a b*)) (*y* (+ *a c*))) (+ *x y*)). In this example, *a* and *c* are bound to 3 and 7, respectively, while *b* is bound to placeholder_102. Partially evaluating the expression requires first partially evaluating (+ *a b*), then (+ *a c*), and then (+ *x y*).**

| Expression | Result | Instruction emitted |
|---|---|---|
| (+ *a b*) | placeholder_243 | placeholder_243 := 3 + placeholder_102 |
| (+ *a c*) | 10 | None |
| (+ *x y*) | placeholder_244 | placeholder_244 := placeholder_243 + 10 |

## An example: Inner product

As an illustration of partial evaluation, consider the vector inner-product program shown here. In this hypothetical application, each input vector is known to contain three floating-point numbers. Furthermore, the numerical value of the last element of each vector is known during partial evaluation. This information is encoded in the input data structures at compile time.

```
(define (inner-product v1 v2) ;;take 2 vectors as arguments
  (let ((length (vector-length v1)))
    (define (inner-product-loop sum counter)
      (if (< counter length) ;;loop through the vector elements
          (inner-product-loop (+ sum
                                 (* (vector-ref v1 counter)
                                    (vector-ref v2 counter)))
                              (+ counter 1))
          sum))
    (inner-product-loop 0 0)))

(define input-vector-1
  (vector (make-placeholder 'floating-point) ;;placeholder #1
          (make-placeholder 'floating-point) ;;placeholder #2
          3.14))

(define input-vector-2
  (vector (make-placeholder 'floating-point) ;;placeholder #3
          (make-placeholder 'floating-point) ;;placeholder #4
          42.0))

(pe vector-inner-product input-vector-1 input-vector-2)
```

When the inner-product program is run during partial evaluation, execution starts with the call to (vector-length v1), which returns 3. This is the first saving provided by partial evaluation: The vector-length call is executed and is not included in the compiled program.

Execution continues with the call to inner-product-loop with sum=0 and counter=0. (vector-ref v1 0) returns <placeholder #1>, and (vector-ref v2 0) returns <placeholder #2>. Again, these vector references are performed during partial evaluation, and will not appear in the compiled program.

```
(* <placeholder #1> <placeholder #3>) ==> <placeholder #5>
```

The multiply cannot proceed during partial evaluation because numerical values for the placeholders are not yet available. A multiply instruction is emitted to perform the multiply at runtime, and a new placeholder, <placeholder #5>, is created to represent the result of the multiply operation.

```
(+ sum <placeholder #5>) ==> <placeholder #5>
```

the placeholders' locations within the compile-time input data structures. Similarly, the compiler automatically creates the epilogue, using the placeholders' locations within the result value *V* produced by partial evaluation.

The compiler also targets the body for a particular machine. Sequential computers usually require reordering of the computation to minimize the number of intermediate results created, thereby minimizing memory accesses. For parallel computers, scheduling is more complicated, requiring that the computation be partitioned among multiple processors.

Figure 5 shows the partially evaluated inner-product example (presented in the sidebar below), with the additional prologue and epilogue sections. For an extremely small program like inner-product, the vector references required to interface to the high-level Scheme program represent a significant cost. However, on larger examples, such as the circuit simulation program discussed next, the compilation process is far more effective. Although high-level data-structure (vector) manipulations remain in the prologue and epilogue, these are insignificant compared with the number of data-structure manipulations (such as manipulation of matrices) that are eliminated through partial evaluation.

The two routines in Figure 6 constitute the inner loop of transient analysis for linear circuits. The function next-state accepts a circuit state at time $t$ and a time increment $h$, and returns the state at time $t + h$. It first calculates the node voltages by creating and solving a sparse matrix. Then the branch currents are computed using the node voltages. The function create-integration-matrix uses object-oriented techniques to add the contributions of each component into the matrix: It retrieves the function for computing an element's contributions from the element itself and then invokes the function. We show these fragments to emphasize the amount of work the simulator must perform to compute the next state.

For the circuit shown in Figure 7, Figure 8 shows how the specialized next-state is compiled. The specialized function maps a state at time $t$ into a state at time $t$ + 0.1. Optimizations that were applied include dead-code elimination, constant folding, sign targeting, and arithmetic simplification. For example, constant folding produced such constants as .02 and 49.6277915633.

Since sum=0, this operation can proceed at compile time, even though the value represented by <placeholder #5> is not yet available.

The inner-product-loop is then called recursively, with sum = <placeholder #5> and counter = 1. The second iteration through the loop creates <placeholder #6> and <placeholder #7> to represent the results of the multiply and the add operations.

During the third iteration through the inner-product-loop, numerical values for the vector elements are available, allowing the multiply to proceed at compile time. The addition is delayed until runtime, creating <placeholder #8> to represent the result of the overall computation. The program produced by the partial evaluator contains no data structures, procedure calls, or conditional tests; there are only numerical operations.

Below is the result of partially evaluating inner-product. The multiplication of 3.14 times 42.0 to produce 131.88 took place during partial evaluation. All vestiges of the original vectors and the inner-product-loop control structure — and portions of the computation — were eliminated by performing them in advance, during partial evaluation.

```
Inputs: Placeholder_1, Placeholder_2, Placeholder_3, Placeholder_4

;;from the first iteration of inner-product-loop:
Placeholder_5 = (* Placeholder_1 Placeholder_3) ;;vector elements #0

;;from the second iteration of inner-product-loop:
Placeholder_6 = (* Placeholder_2 Placeholder_4) ;;vector elements #1
Placeholder_7 = (+ Placeholder_5 Placeholder_6) ;;compute sum

;;from the third iteration of inner-product-loop:
Placeholder_8 = (+ Placeholder_7 131.88)

Result:
Placeholder_8
```

Traditional compiler optimizations further improve the performance of the partially evaluated program. Algebraic simplification, dead-code elimination, and common subexpression elimination optimize the underlying numerical computation, without interference from compound data structures or abstraction mechanisms. Opportunities for these optimizations often arise when high-level data-structure operations are combined, as in this version of the subtract-vectors operation, where symbolic manipulation of the low-level computation allows the addition and scaling operations to be combined in a subtraction.

```
(define (subtract-vectors a b)
        (add-vectors a
                     (scale-vector -1 b)))
```

Such optimizations are often not noticed by the programmer when the optimizations do not apply uniformly to all elements of a data structure, or when the operations being combined are in physically separate portions of the program.

```
Compiled INNER-PRODUCT, Arguments: v1, v2

;;PROLOGUE:
Placeholder_1 = (vector-ref v1 0)
Placeholder_2 = (vector-ref v1 1)
Placeholder_3 = (vector-ref v2 0)
Placeholder_4 = (vector-ref v2 1)

;;BODY:
;;from the first iteration of inner-product-loop:
Placeholder_5 = (* Placeholder_1 Placeholder_3)
;;vector elements #0

;;from the second iteration of inner-product-loop:
Placeholder_6 = (* Placeholder_2 Placeholder_4)
;;vector elements #1
Placeholder_7 = (+ Placeholder_5 Placeholder_6) ;;sum

;;from the third iteration of inner-product-loop:
Placeholder_8 = (+ Placeholder_7 131.88)

;;EPILOGUE:
Placeholder_8
```

**Figure 5. Compiled inner-product program with the additional prologue and epilogue instructions required to interface to high-level Scheme programs.**

```
(define (next-state circuit state h)
  (let* ((matrix (create-integration-matrix circuit state h))
         (new-voltages  (solve-matrix
                          (trim-ground matrix)))
         (new-currents (compute-b-currents circuit
                          new-voltages state h)))
     (make-circuit-state new-voltages new-currents
                          (+ h (state-time state)))))

(define (create-integration-matrix circuit state h)
  (let ((voltages (state-voltages state))
        (currents (state-currents state)))
    (let loop ((components (circuit-components circuit))
               (matrix (create-nxn+1-matrix
                          (circuit-number-of-nodes circuit))))
      (if (null? components)
          matrix
          (loop (cdr components)
                ((component-integration-method
                  (car components))
                  matrix voltages currents h))))))
```

**Figure 6. Scheme code fragments for transient analysis.**

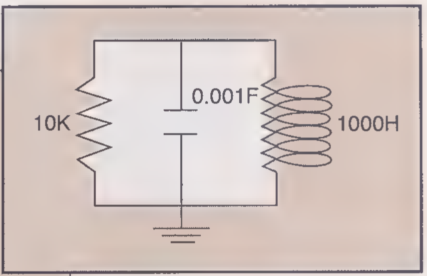


**Figure 7. Circuit for Figure 8.**

```
(compile next-state
  rlc-circuit
  (make-circuit-state (voltages (make-placeholder 'floating-point))
                      (currents (make-placeholder 'floating-point)
                                (make-placeholder 'floating-point)
                                (make-placeholder 'floating-point))
                      (make-placeholder 'floating-point))
  0.1)

Compiled NEXT-STATE, Arguments: state

;;PROLOGUE:
Placeholder_1 = (vector-ref (vector-ref state 0) 0)
temp = (vector-ref state 1)
Placeholder_2 = (vector-ref temp 1)
Placeholder_3 = (vector-ref temp 2)
Placeholder_4 = (vector-ref state 2)

;;BODY:
Placeholder_6 = (* Placeholder_1 .00005)
Placeholder_8 = (+ Placeholder_6 Placeholder_2)
Placeholder_9 = (* Placeholder_1 .02)
Placeholder_10 = (+ Placeholder_9 Placeholder_3)
Placeholder_12 = (- Placeholder_10 Placeholder_8)
Placeholder_14 = (* Placeholder_12 49.6277915633)
Placeholder_15 = (- Placeholder_14 Placeholder_1)
Placeholder_17 = (* Placeholder_15 .02)
Placeholder_18 = (- Placeholder_17 Placeholder_3)
Placeholder_19 = (+ Placeholder_14 Placeholder_1)
Placeholder_21 = (* Placeholder_19 .00005)
Placeholder_22 = (+ Placeholder_21 Placeholder_2)
Placeholder_23 = (* Placeholder_12 4.96277915633e-3)
Placeholder_24 = (+ .1 Placeholder_4)

;;EPILOGUE:
(VECTOR (VECTOR Placeholder_14)
        (VECTOR Placeholder_23 Placeholder_22 Placeholder_18)
        Placeholder_24)
```

**Figure 8. Compiling next-state for Figure 7 circuit and a fixed time step. The compiler accepts a function and the partial values that describe the function's inputs.**

Table 2. **Timings of the sample applications (in seconds) and speedups with partial evaluation (in percent). For the *n*-body problem, both the time step and the masses of the planets were chosen at compile time.**

| Problem description | Compiled C Scheme | Specialized program | Speedup over compiled |
|---|---|---|---|
| 6-body system | 0.76 | 0.020 | 38 |
| 9-body system | 1.50 | 0.038 | 39 |
| Translate $P = 3$ | 0.022 | 0.002 | 11 |
| Translate $P = 6$ | 0.28 | 0.011 | 25 |
| Duffing's equation | 4.04 | 0.53 | 7.6 |
| Circuit simulation | 2.37 | 0.026 | 91 |

## Limitations of partial evaluation

Partial evaluation works best when the structure of the system stays constant and only the state changes. Simulations of circuits, dams, and solar systems fall into this class. It does not work well when the structure changes or the computations are extremely data dependent. For example, partial evaluation does not work for sorting arrays or inserting elements into balanced trees. Similarly, it is difficult to use with linear programming, because the choice of pivot is data dependent.

A program must be partially evaluated whenever the structure of the problem changes. This is not a drawback with simulations that run for a long time or with applications such as circuit simulation, where multiple sets of input data and initial conditions need to be run before the system structure is changed. However, with smaller problems, where specialized code is not traversed hundreds of times, the time spent in the partial evaluator may exceed the time saved by the optimizations.

The size of the compiled program can become a problem because loops are expanded at partial evaluation time. Very large data sets and nonlinear algorithms result in very large specialized programs. When the code becomes too long, selected data structures and loops should be left intact. For example, the inner loops that deal with manipulations of a single segment of a large data structure can be partially evaluated, while the outer loop that traverses the data structure can be left intact. Our compiler leaves to the programmer the choice of which loops to partially evaluate, although the decision could be automated by the proper heuristics.

Requiring the programmer to decide which regions of a data-dependent program to partially evaluate is a limitation of our technology. Much of the partial evaluation community is investigating automatic methods that do not require programmer intervention to handle data-dependent programs. Their technologies and methods for full automation have achieved many successes and are getting more powerful, but are not yet able to handle the types of programs and programming styles that our partial evaluator can handle.

## Experiments

We have applied partial evaluation to several numerically oriented scientific problems. These problems were chosen from active research at MIT and Stanford, providing a "real world" demonstration of partial evaluation's applicability to scientific computation. Scheme programs implementing the *n*-body algorithm, the solution to Duffing's equation, the translation operator for the multipole method, and an electrical circuit simulator were taken directly from code in use by researchers.

The figures presented here measure performance using C syntax programs produced by the compiler. The application programs were not modified for these experiments, except for the Duffing's equation application, in which a programmer's declaration was added, indicating that the main integration loop should be left intact.

The experimental method was as follows:

(1) Obtain working code from researchers.

(2) Select the parts of the code to be partially evaluated.

(3) Compile the selected code with our compiler and produce a C program as output.

(4) Compile the C program with a conventional compiler and link it into the MIT-Scheme Lisp system, so that it can be invoked as a subroutine from Lisp.

(5) Compile the program using a conventional Lisp compiler. (Specifically, MIT C Scheme Release 7 with Liar compiler Version 4.38, running on a Hewlett-Packard 9000, Series 350, with 16 Mbytes of memory. The timings presented do not include garbage collection time.)

(6) Compare the execution times of the conventionally compiled program with those of the partially evaluated program.

**Applications.** We applied this method to four applications: the *n*-body problem, the multipole method translation operator, Duffing's equation, and an electrical circuit simulation.

*The* n*-body problem.* The *n*-body problem involves computing the trajectories of a collection of *n* particles that exert forces on each other. This very important problem arises in particle physics, astronomy, and space travel. In astronomy, the six-body and nine-body problems are of particular interest. The six-body problem includes only the outer planets and the sun for investigations of the long-term stability of the solar system. The nine-body problem includes all the planets except Mercury, which is excluded because its high eccentricity necessitates an extremely small integration-step size, making long-term integrations impractical.

An *n*-body program written in Scheme by Gerry Sussman was used as a starting point for the compilation process. This program makes liberal use of abstraction mechanisms, including higher order procedures, lists, vectors, table lookups, and set operations.

To simulate future particle motion, the program integrates the forces that the particles exert on each other over time. The integration-step routine takes an initial state of the planets and produces a new state that corresponds to one time step later. This routine is then repeated, thereby advancing the system in time. We used our compiler to create a specialized version of the integration-step procedure.

The state of the system includes the planets' positions, velocities, and masses. The data description presented to the compiler left the positions and velocities unknown, but specified the masses, which are virtually time independent. Many computations involving the planets' masses were performed at compile time. For example, since Pluto is very small relative to the other planets, its mass was approximated as zero. The partial evaluator propagated this information throughout the program, eliminating numerous computations.

For a given $n$, the $n$-body problem is entirely data independent. Measurements were taken for the six-body problem and for the nine-body problem, using the Runge-Kutta integration method. We found that when the masses of the planets were provided at compile time, the partially evaluated programs ran 11 percent faster than if the masses of the planets were not known until runtime.

*The multipole method translation operator.* The multipole method approximates force interactions involving large numbers of particles, as in fluid-flow simulations. The method divides space into a quadtree-like tree of cubes. Part of the force approximation propagates information up the tree from a cube to its parent. A significant portion of the computation time is spent evaluating translation operators. The translation operator is an entirely data-independent computation.

We took a Scheme implementation of this operation from a program written primarily for people to understand. As such, the program does not take advantage of special cases in the multipole expansions, such as terms that are known to have exponents of 0 or 1. Experiments showed that roughly half the numerical operations were eliminated because of algebraic simplification involving these constants. The program was compiled for two different values of a parameter $P$, which denotes the number of terms in the multipole expansions. ($P = 3$ is commonly used for benchmark purposes. For large $P$ — above 10 — the growth in code size makes compilation of the entire translation operator impractical. For such large $P$, either a smaller segment could be compiled or some loops could be left intact.)

*Duffing's equation.* To demonstrate the compilation of programs containing simple loops, an adaptive Runge-Kutta integrator was used to integrate a one-period evolution of the variations and derivatives of Duffing's equation. This program was taken from work on automatically characterizing the state space of Duffing's equation.[4] It uses an adaptive integration strategy coupled with a control loop that iterates for one period. A declaration was added to the program, telling the partial evaluator not to try to unroll the control loop.

*Electrical circuit simulation.* Partial evaluation was applied to an electrical circuit simulator implemented in Scheme. This simulator was written abstractly to reflect as much of the underlying mathematics of simulation as possible. Abstract structure allows experimentation with different simulation algorithms and strategies. We used partial evaluation to specialize this simulator for the circuit of interest, providing a dramatic performance improvement. The experiment we performed simulated a 120-component linear circuit; the integration time step was not specified until runtime.

**Performance measurements.** Our compiler generated specialized routines in C for each of the applications described above. Table 2 presents timings and speedup factors for each application, compiled by the Liar Scheme compiler ("compiled C Scheme"), and compiled by our partial-evaluation-based compiler ("specialized program"). None of these timings includes the time required to compile the specialized C routines themselves. The specialized routines are significantly faster than the Scheme programs they were generated from. For abstract programs, specialization provides dramatic performance improvements.

The performance of our compiler itself has not been investigated. For our experiments, partial evaluation time ranged from tens of seconds to several minutes (all programs and timings were run on the same hardware platform). A problem in performing measurement experiments was compiling the specialized programs with a C compiler. The huge basic blocks that appear in specialized programs break many C-code optimizers: The optimizers do not seem to terminate. This problem can be solved by generating machine code directly, a task we have not yet pursued.

## Mapping programs onto parallel architectures

Partial evaluation exposes tremendous amounts of instruction-level parallelism. This is very important, as the effective use of superscalar and superpipelined processors often requires program transformations to expose the parallelism needed to keep them completely busy.[5] The first author implemented several analysis and scheduling programs to study and harness this parallelism. For a hypothetical architecture consisting of multiple arithmetic logic units and a communication network, experiments were run to measure the effects of pipeline and communication latencies on performance. At least for the nine-body problem, large numbers of arithmetic logic units could be kept continuously busy, thereby efficiently harnessing the available parallelism. (Specifically, the problem was 12th-order Stormer integration of the nine-body gravitational attraction problem, with masses chosen at compile time and time step chosen at runtime.)

The first step in these experiments was to construct a directed acyclic graph from the body of the partially evaluated program. Each node in the graph represents an operation, and there is a directed edge from the producer of a value to each of the consumers of the value. (Actually, the graph was created incrementally by the partial evaluator as it constructed the body.) We call the directed acyclic graph a numerical dataflow graph.

Figure 9 presents a parallelism profile for Stormer integration of the nine-body problem. This profile shows the maximum amount of parallel execution that would occur if a computer had an infinite number of processors communicating instantaneously. The profile was produced by performing a breadth-first search of the numerical dataflow graph, scheduling each operation as soon as it could be performed.

This profile differs from the parallelism profiles common in the literature in that it accounts for the different latencies of the different arithmetic operations. (The latencies were based on Bipolar Integrated Technologies' B3110A/B3120A floating-point chips.) We discovered that for double-precision computations, latency differences are large enough to be of fundamental importance. For our realistic latency measures, the critical path length differs by a factor of 2 when we account for latencies.

**Architectural constraints that increase latency.** Pipelining and communication delays interfere with efficient execution of numerical dataflow graphs, increasing the effective time required to complete an operation. In pipelining, several instructions are executed simultaneously within a processor. Pipeline latency is the number of cycles required for the result of an operation to become available as the source of another operation. Communication latency is the number of cycles required to transfer a result between processors.

*Pipelining*. Technological considerations often result in pipelined architectures that overlap the execution of successive instructions within a single processor. The parallelism profile in Figure 9 is based on the assumption that the result of an instruction that finishes executing in one cycle can be used immediately in the following cycle. Unfortunately, this assumption is not valid with pipelining. Figure 10 shows that in a three-stage pipeline the result of an instruction initiated in cycle 1 will not be available to the instruction initiated during cycle 2. Thus, even with an infinite number of processors and no communication delays, a machine composed of three-stage pipelined processors will require about twice as many cycles to execute a computation as a nonpipelined machine would.

Since some instructions have more latency than others, the processors are sometimes busy more than half the time. This would make "twice as many cycles" seem too pessimistic. On the other hand, the estimate does not consider the additional delay imposed by unloading a result from a processor before it can be loaded into another processor. This creates a one-cycle cost for moving data between processors, even when there are no communication delays, effectively increasing the minimum number of cycles required to complete the computation. Overall, these two effects cancel each other out.

Despite this increase in the number of cycles required to execute a program, pipelining is advantageous because it reduces the length of each cycle. In addition, parallelism available in the problem can be used to hide the latency imposed by pipelining. Rather than schedule all available parallel operations into the same cycle on many processors, it is possible to use fewer processors more effectively, scheduling some of the operations during the next cycle (parallelism in time) to keep the pipeline busy.

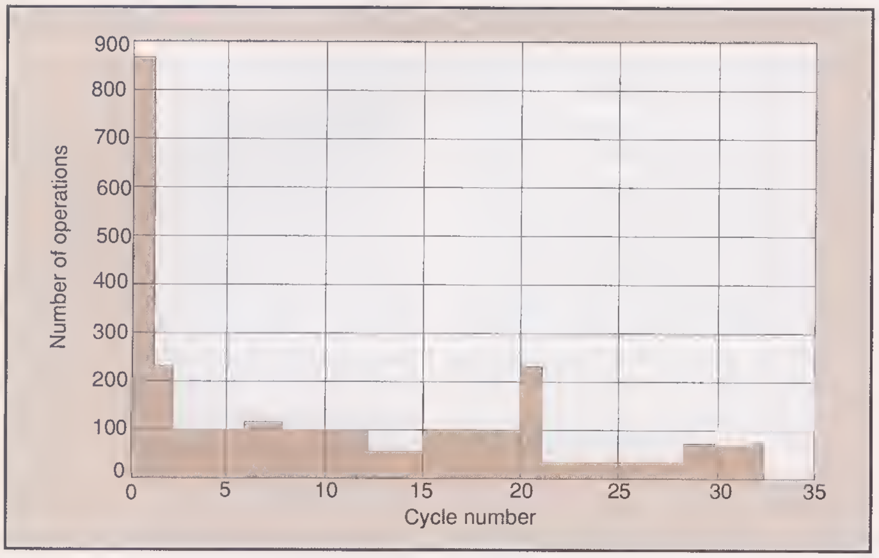


**Figure 9. Parallelism profile of the nine-body problem. This graph represents the total parallelism available in the problem, accounting for the latency of numerical operations.**

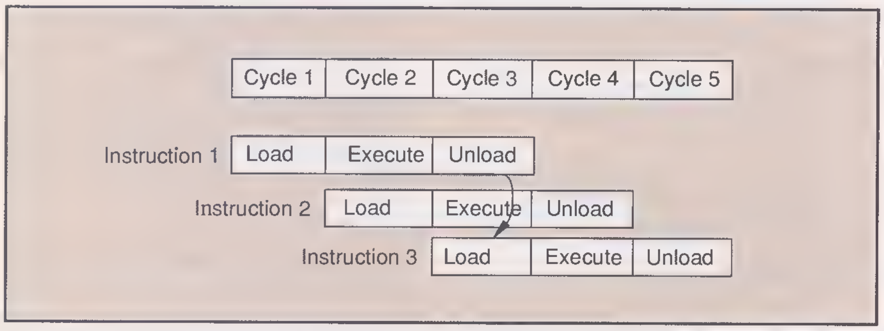


**Figure 10. A typical three-stage processor pipeline. During the load stage, the data is loaded into the arithmetic logic unit. The result is computed during the execute stage, and unloaded from the ALU during the unload stage. The results produced by instruction 1 are not available to instruction 2, but are available to instruction 3.**

*Communication latency*. Processors do not communicate instantaneously. The time required to move a result from one processor to another limits how soon the result can be used by a subsequent instruction. This has an effect similar to increasing the length of the pipeline, as Figure 11 illustrates. Just as parallelism can be used to hide the latency in pipelines, parallelism can also hide the latency imposed by communication delays.

**A scheduler for parallel programs.** Our scheduler searches for a schedule that will keep each processor as busy as possible. It uses heuristics that spread the available parallelism over the processors to hide the latencies imposed by pipeline and communication delays. These heuristics schedule the critical path first and schedule noncritical operations around the critical path. For the nine-body problem, the system was able to use 40 pipelined processors with 90 percent efficiency.

The scheduler operates on the numerical dataflow graph. It first computes the latency of every possible path through the graph. These paths are then sorted, allowing the critical path of the computation to be identified. When the operations are scheduled, priority is given to operations that lie in the critical path of the computation. If all available processors are not needed to work on the most critical path, computations from less critical paths are scheduled.

Depending on the machine model, creating an optimal schedule that completes in the shortest time possible can be an NP-complete problem. Rather than try to find an optimal solution to the problem, heuristics are used to select a solution that keeps the processors extremely busy.

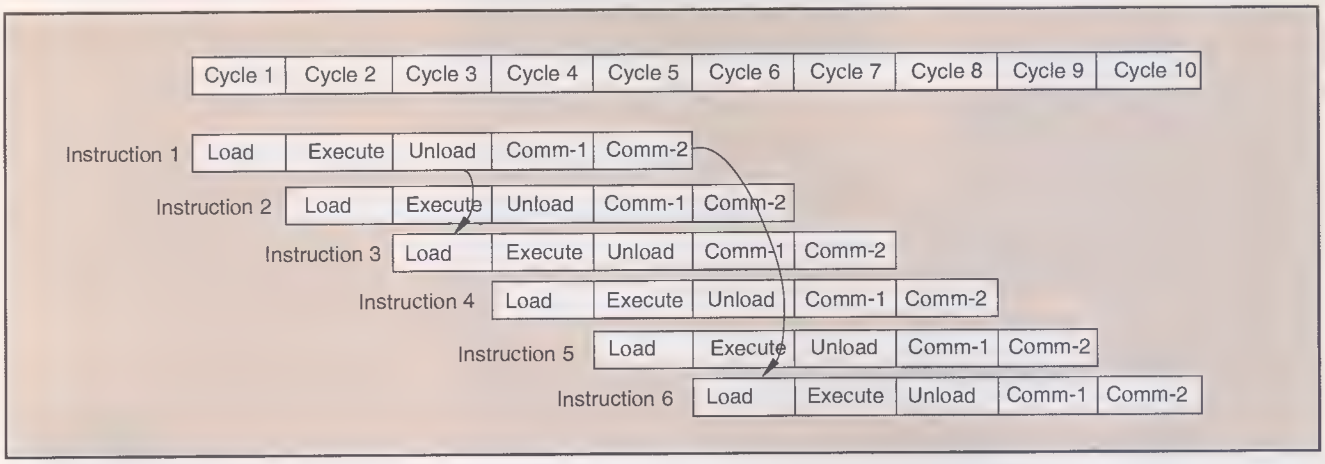


**Figure 11. A three-stage processor pipeline with a communication latency of two cycles. As indicated by the arrows, a result produced hy instruction 1 can be used within the same processor hy instruction 3, but cannot be used by other processors until instruction 6.**

To give a flavor for the algorithm and heuristics, here is a brief overview:

- A subset $O$ of the operations whose operands have been computed is chosen, corresponding to the number of processors available. This selection is based on the latency priorities described above.
- The operations in $O$ whose operands have been available long enough to have been transmitted to other processors are given lower scheduling priority than those operations whose operands have been produced recently. This rule gives priority to nonrelocatable computations.
- A computation whose operands were produced by a processor will be scheduled in that same processor wherever possible.
- The number of connections between processors is kept to a minimum. When the operands of a computation must be transmitted from one processor to another, the scheduler attempts to choose a pair of processors that have communicated with each other before.
- Several heuristics break ties, using such information as the memory usage within each processor, the number of computations waiting for a particular result, and the frequency with which processors use the communication network.

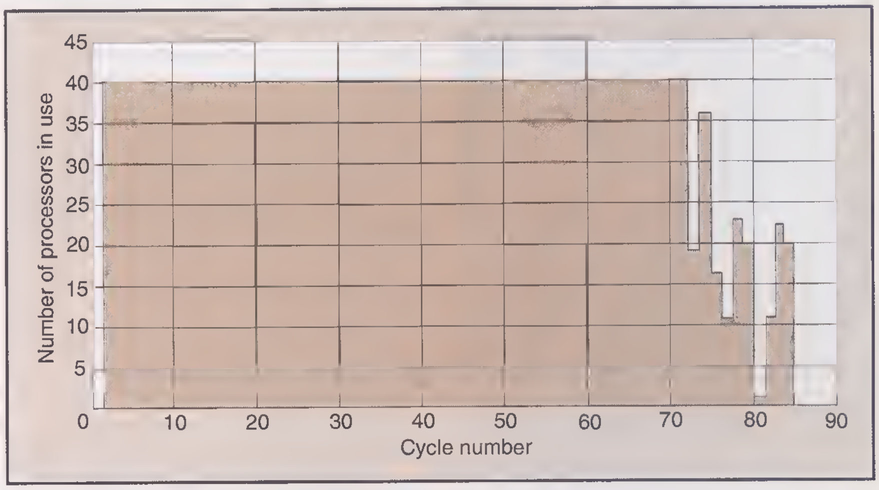


**Figure 12. The result of scheduling the nine-body problem onto 40 pipelined processors with a communication latency of one cycle. A total of 85 cycles was required to complete the computation. On average, 36.4 of the 40 processors were used during each cycle.**

These heuristics are quite effective.[6] On the nine-body problem, the scheduled code provided speedups approaching the theoretical limit.

**Performance measurements.** Figure 12 shows the results of applying the scheduler to the nine-body problem, using a 40-processor system with a three-stage processor pipeline and a communication latency of one cycle. The parallelism available in the problem was distributed over the life of the computation, effectively using all 40 processors in most of the cycles. Overall, the performance improved 36-fold over that of a single pipelined processor, indicating that the processors were used with approximately 90 percent efficiency.

The scheduler's ability to use the available processors effectively varies with both the number of processors in the system and the communication latency. As Figure 13 shows, for the nine-body problem we found that communication latency directly affects the maximum speedup provided by the scheduler.

**Relation to other parallelization research.** Many compilers for high-performance architectures use program transformations to exploit low-level parallelism. For instance, compilers for vector machines unroll loops to help fill vector registers. Similarly, compilers for very-large-instruction-word architectures[7] use trace scheduling to guess which way a

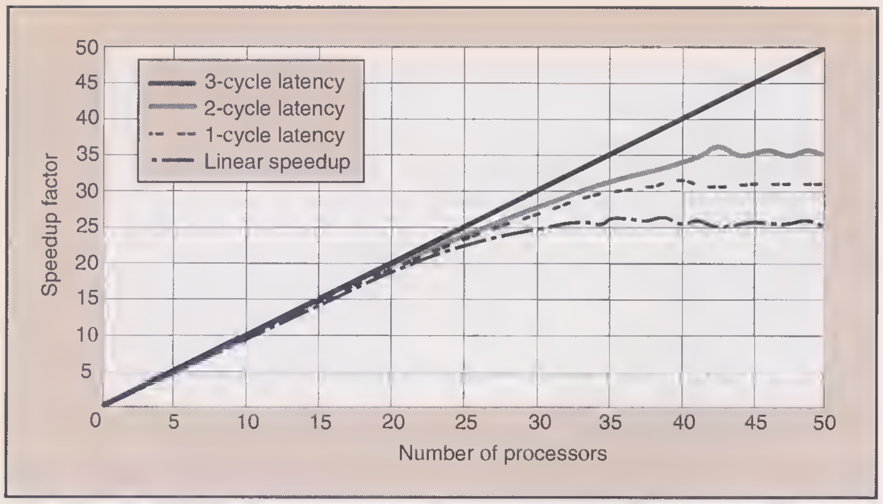


**Figure 13. Effects of communication latency on speedup. The graph shows the speedup factors over a single pipelined processor. The analysis is for a system composed of processors using a three-stage pipeline.**

branch will go, allowing computations beyond the branch to occur in parallel with those that precede the branch. These techniques are limited by their preservation of the original program's user data structures: If the original program represented an object as a vector of vectors, the compiled program will do so as well. Preserving data structures imposes synchronization requirements that reduce the instruction-level parallelism available to the compiler.

Partial evaluation eliminates data structures and many conditionals to produce numerical dataflow graphs, allowing intermediate results to be used in portions of a program that would not otherwise have been reached, even through trace scheduling. This technique is orthogonal to the trace-scheduling approach: Partial evaluation eliminates conditional tests related to data structures, producing large data-independent regions (also known as basic blocks) that can be executed in parallel, while trace scheduling optimizes across basic block boundaries.

Partial evaluation is an important technique that provides significant performance improvements for an important class of numerical programs. Implementing partial evaluation using the placeholder technique is adequate for data-independent computations, but it needs to be made more general, particularly in the area of automatically deciding which loops and data structures should be specialized and which should be left for runtime evaluation.

The most exciting result of this work is the ability of partial evaluation to make abstractly specified programs execute efficiently. One of the most frustrating tasks in scientific programming is transforming an application to a form that makes use of existing library routines. Partial evaluation will allow library routines to be specialized to match the program, rather than requiring the programmer to transform the program to match the library routine. ■

## Acknowledgments

This article describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology and at the Computer Systems Laboratory of Stanford University. The MIT AI Laboratory research is supported in part by the Advanced Research Projects Agency of the Defense Department under ONR contract N00014-86-K-0180. The Computer Systems Laboratory research is supported in part by contract N00014-87-K-0828. Daniel Weise is also supported by NSF contract MIP-8902764.

We would like to thank Feng Zhao, Harold Abelson, Gerry Sussman, Erik Ruf, Scott Seligman, Thomas Simon, and Olivier Danvy for helping us to build the systems and coherently report the results and major ideas. We also thank the anonymous referees who supplied us with many useful and constructive comments.

## References

1. G. Roylance, "Expressing Mathematical Subroutines Constructively," *Proc. ACM Conf. Lisp and Functional Programming*, ACM, New York, 1988, pp. 8-13.
2. M. Halfant and G.J. Sussman, "Abstraction in Numerical Methods," *Proc. ACM Conf. Lisp and Functional Programming*, ACM, New York, 1988, pp. 1-7.
3. A. Aho, R. Sethi, and J. Ullman, *Compilers: Principles, Techniques, and Tools*, Addison-Wesley, Reading, Mass., 1985.
4. H. Abelson, "A Step Towards the Automatic Analysis of Dynamical Systems," Memo 1174, Artificial Intelligence Laboratory, MIT, Cambridge, Mass., 1989.
5. N. Jouppi and D. Wall, "Available Instruction-Level Parallelism for Superscalar and Superpipelined Machines," *Proc. Third Internal Conf. Architectural Support for Programming Languages and Operating Systems*, 1989, pp. 272-282.
6. A. Berlin, *A Compilation Strategy for Numerical Programs Based on Partial Evaluation*, masters thesis, MIT; also Tech. Report TR-1144, Artificial Intelligence Laboratory, MIT, Cambridge, Mass., 1989.
7. J.R. Ellis, *Bulldog: A Compiler for VLIW Architectures*, MIT Press, Cambridge, Mass., 1986.

## Further reading

Partial evaluation has many uses and is becoming an active research topic. The first partial evaluators were written by the artificial intelligence community for Lisp. Their motivation was much like ours: to remove the costs of abstraction automatically. Kahn has written an excellent paper on the value and application of partial evaluation to artificial intelligence research.

Researchers at the University of Copenhagen have investigated partial evaluation to construct compilers automatically from denotational descriptions of programming languages. Jones, Sestoft, and Sondergaard report some of their progress.

An excellent partial evaluation source book is *Partial Evaluation and Mixed Computation*, edited by Bjorner, Ershov, and Jones. This book also contains a comprehensive bibliography of partial evaluation research.

Chambers and Ungar have used partial evaluation techniques to efficiently compile Self, an object-oriented language. They specialize programs on the fly, as specialized (they use the word "custom") routines are needed. This work places a heavy emphasis on quickly producing the specialized procedures.

Bjorner, D., A.P. Ershov, and N.D. Jones, eds., *Partial Evaluation and Mixed Computation*, North-Holland, Amsterdam, 1988.

Chambers, C., and C. Ungar, "Customization: Optimizing Compiler Technology for Self, a Dynamically-Typed Object-Oriented Programming Language," *Proc. SIGPlan Conf. Programming Language Design and Implementation*, ACM, New York, 1989, pp. 146-160.

Jones, N.D., P. Sestoft, and H. Sondergaard, "Mix: A Self-Applicable Partial Evaluator for Experiments in Compiler Generation," *Int'l J. Lisp and Symbolic Computation*, Vol. 1, Nos. 3/4, 1988.

Kahn, K.M. "A Partial Evaluator of Lisp Programs Written in Prolog," *First Int'l Logic Programming Conf.*, Marseilles, France, 1982, pp. 19-25.

**Andrew Berlin** is a PhD student in electrical engineering and computer science at MIT. His research interests are in computer architecture, parallel compilation, and scientific applications of computer technology.

Berlin received his BS and MS in electrical engineering and computer science from MIT in 1985 and 1989.

**Daniel Weise** is an assistant professor in the Computer Systems Laboratory at Stanford University. His research interests include programming languages, compilers, program transformation, and computer-aided design.

Weise received his BS from UCLA in mathematics and computer science in 1979 and his MS and PhD in computer science from MIT in 1982 and 1986. He is a member of the IEEE Computer Society, IEEE, and ACM.

Berlin can be reached at the Artificial Intelligence Laboratory, MIT, 545 Technology Square, Cambridge, MA 02139. Weise can be contacted at the Computer Systems Laboratory, Stanford University, Stanford, CA 94305.

SCHOOL OF COMPUTER SCIENCE AND INFORMATION SYSTEMS

## What's so special about Frances Gustavson?

Dr. Frances Gustavson began her career at IBM's T.J. Watson Research Center in 1964. Since then, she's held several key positions in the industry, including first chair of Pace University's Westchester Information Systems Department, and has been a consultant in the industry. Those 26 years of experience give her a special insight when it comes to teaching.

"When experienced professionals return to the classroom it's because they want to further their career. Pace offers a range of programs that have been specially created with that person in mind.

"Pace programs have been designed to give you experience and expertise that go far beyond your degree: how to think; how to apply what you know to solve problems; and how to continue growing professionally once you've left the classroom.

"Our Master of Science degree in Computer Science is one of the only programs of its kind offered in the New York/Westchester area. It provides a broad education in this exciting and dynamic field with a strong focus on software engineering and development.

"The Master of Science degree in Information Systems is another first for Pace. Most schools offer this kind of coursework only as part of an MBA program. But this was specially created for people who want to develop ways to integrate computers into modern organizations.

"Computers are playing an ever increasing role in education today. Our Certificate in Computer Science for Teachers is for educators who want to keep up with the latest innovations and want to integrate computers into their classrooms."

**How to take the first step.**

Put Frances Gustavson's experience to work for you. Fill out this coupon and mail it today. Or call 1-800-TRI-PACE Ext. I001. Do it now. The experience for a lifetime is waiting for you.

**Mail to: Pace University, Admissions Information Center, Pace Plaza, NY, NY 10038**

Yes, I'm interested in attending an Information Session on:
☐ White Plains Graduate Center, 12-3-90
☐ Please send me a catalog and application for the School of Computer Science & Information Systems
I am interested in attending:
☐ Full-time ☐ Part-time ☐ Westchester ☐ New York

Name: ____________
Address: ____________
City/State/Zip: ____________
Phone: ____________
Area of interest: ________ SS#(opt.): ________

4MO1001

PACE UNIVERSITY

THE EXPERIENCE FOR A LIFETIME

# Architecture-Independent Parallel Computation

David B. Skillicorn
Queen's University at Kingston

**Locality-based computation, the foundation for an architecture-independent programming language grounded in the Bird-Meertens formalism, shows that architecture-independent parallel programming is possible.**

Parallel computers have failed to make a major impact on mainstream computation, despite the fact that commercial products have been available for almost a decade. A substantial performance/price advantage over conventional supercomputers, and even large uniprocessors, has not been enough to convince users to move from a sequential to a parallel mode of computation.

An examination of the state of the art in parallel computing suggests an explanation. Different classes of parallel architectures require radically different paradigms for describing and executing computations. In addition, both practitioners and theoreticians have specialized along architectural lines. There is no obvious winner among these architectures; it is hard to move applications from one class to another; and many potential users are unwilling, on the present evidence, to make a computer acquisition decision with long-term implications.

There is currently no way to develop software for parallel computers and expect it to have a long lifetime. Software developed for uniprocessors has turned out, rather surprisingly, to have a very long lifetime indeed. A great deal of software that was written more than 20 years ago is still in use. By contrast, developers of software for parallel computers do not expect their software to have a very long life span; they are often resigned to substantially reworking their programs with the advent of the next generation of computers.

Existing parallel languages are almost all tied to some particular architectural class. Even when the software environment seems superficially the same (some variant of C or Fortran, perhaps), the underlying mechanisms for communication and synchronization are often substantially different. In some cases, software must be substantially modified to take full advantage of, or even to execute on, a larger configuration of the same kind of multiprocessor. Such software is not portable in any serious sense.

Because the paradigms and patterns of program execution for various parallel architectures differ, programmers today must approach parallel programming in ways that are architecture dependent. The standard repertoires of algorithms and program fragments for each of the various architecture classes have very little in common. Moving from one architecture class to another very often means learning to design and program all over again. Thus, programmers are no more "portable" than software.

Software engineering techniques for developing parallel programs have not yet been developed. The present generation of languages requires programmers to be aware of, and explicitly handle, either the degree of physical parallelism, or communication, or both. Programmers must be aware of the kind of architecture on which their software will run, and often the number of processors, their storage capacity, and their configuration. Formal techniques for managing the development of software in this environment must necessarily be complex.

The most popular approach to software engineering for uniprocessors is to begin with a set of requirements, develop a program, and then show that the resulting program satisfies the requirements. The structure of the program is not implicit in the requirements, and it is the programmer's job, using a repertoire of techniques and experience, to decide how the requirements might best be met. It is questionable whether, even in the sequential case, this approach is better than a transformational one, in which programs are derived by algebraic or algorithmic transformation from their specifications. In a parallel environment, the transformational approach seems much better suited, since we do not already have a repertoire of standard techniques, and proofs of requirement satisfaction are harder to obtain.

Another major problem with the current state of parallel computing is the lack of a theory that relates the complexity of algorithms to the complexity of programs running on actual machines. We have forgotten how deeply we make use of the fact that Turing machines are universal. Therefore, an implementation of an algorithm on one manufacturer's uniprocessor will differ in speed by no more than a constant factor from that on another's. In the parallel world, we have no such guarantees. The most popular complexity models, the PRAM (Parallel Random Access Machine) model and the Boolean circuit model, both omit important properties of physical architectures. The result is that prospective purchasers of a specific parallel computer must face the fact that their intended applications may run slower than benchmarks by a nonconstant factor. Should the prospective purchasers decide to buy a particular style of computer, they won't be assured that a new development might not bring a new computer to market that would be better than the existing one by a nonconstant factor. It is no wonder that users have, by and large, held back from buying parallel computers.

The lack of a relevant complexity theory has also made it difficult to assess exactly how much progress has been made in algorithm design. Algorithms developed for different machine classes cannot easily be compared, and the point at which a real improvement has occurred is not always clear.

Progress in bringing parallel computing into the mainstream can only be made by addressing all of these issues. There is some urgency about the problem. For the time being, the speed of uniprocessors continues to increase, parallelizing compilers make it possible to exploit some parallelism in existing sequential software on computers with moderate parallelism, and there is a vast amount of existing sequential software. We expect that these factors will allow the current pattern of sequential software development to continue for a few years. However, it seems likely that both hardware improvements and parallelizing compiler improvements will be subject to diminishing returns. In addition, the continuing high cost of uniprocessors relative to parallel computers will force software developers to change to an environment that can capture substantial parallelism from the start.

Experience suggests that when such a switch occurs, the first viable approach will quickly become the standard. It is important that it should be the right one — one that can provide a growth path for software development for many years. One of the challenges facing computer science researchers is to develop this approach.

In this article, I will consider four major parallel architecture classes:

- single instruction, multiple data or SIMD computers,
- tightly coupled multiple instruction, multiple data or tightly coupled MIMD computers,
- hypercuboid computers, and
- constant-valence MIMD computers.

Other, more specialized, architectures are possible, but the four classes listed above cover all general-purpose parallel computers.

A SIMD computer (see Figure 1) consists of a single instruction processor that broadcasts each instruction to a set of data processors. Each data processor has its own memory and is connected by a switch to the other data processors. Thus, a single instruction stream acts on a large number of data streams. The important characteristic of this architecture class is that only one action can take place at a given time. Even coding instructions as data and triggering them from the instruction processor can't significantly weaken this restriction, as I will demonstrate.

A tightly coupled MIMD computer (see Figure 2) consists of a set of processors

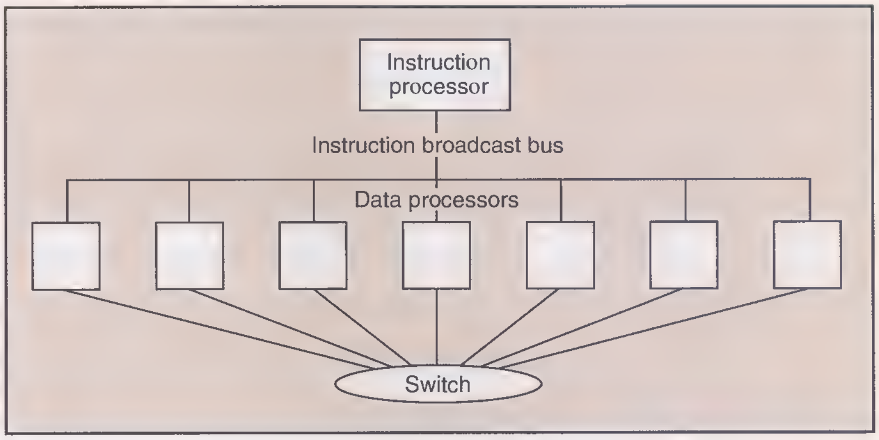


**Figure 1. A single instruction, multiple data computer.**

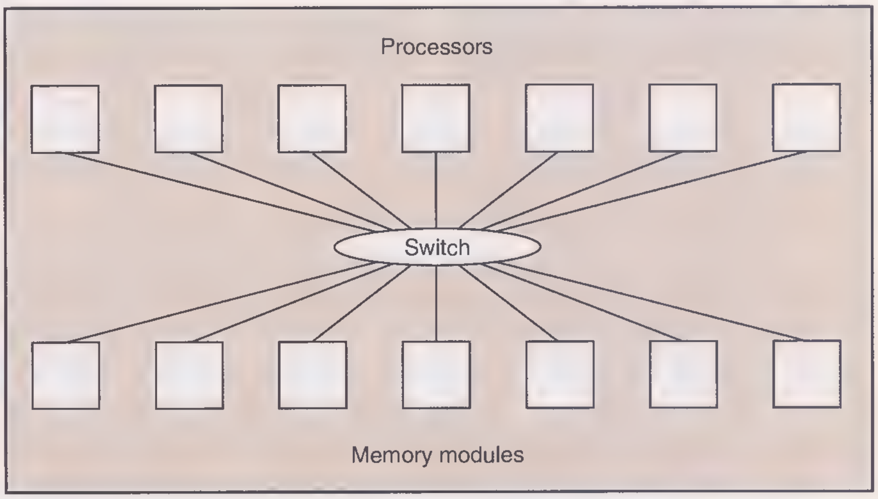


**Figure 2. A tightly coupled multiple instruction, multiple data computer.**

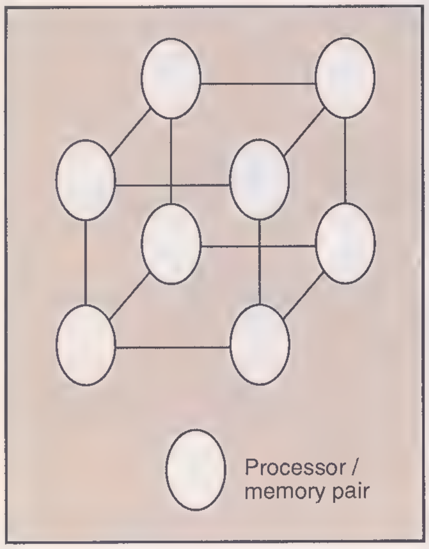


**Figure 3. A hypercuboid MIMD computer.**

connected to a set of memory modules by a switch. Each processor can execute its own thread of instructions, either synchronously with the other processors or asynchronously, and can access any memory location through the switch. Processors cannot communicate with each other except by writing to locations that can then be read by others. The important property of this architecture class is that, at least until optical technology has developed further, there is considerable latency in the switch. If the number of processors is $p$, then the switch depth (and, hence, latency) is $\Omega(p)$. Attempts by more than one processor to read from the same location (in fact, a location in the same module) will fail.

A hypercuboid computer (see Figure 3) is loosely coupled, that is, it consists of processor/memory pairs connected by a communication network. Each processor controls its local memory and can only access a location in the memory of another processor by requesting it to read the value and communicate it, or by sending it a value and asking for it to be stored. Hypercuboid architectures have a communication topology in which the number of links per processor grows as the logarithm of the number of processors in the computer. The hypercube is the best known example in this class. The diameter (that is, the number of links that a message must traverse between most distant processors) is logarithmic in the number of processors.

A constant-valence MIMD computer (see Figure 4) is like a hypercuboid computer, except that the number of links per processor is a small constant. Similarly, the diameter of the communication network is logarithmic in the number of processors. The chief difference between this class and the hypercuboid is the restricted capacity for communication caused by the sparsity of communication links. As I will show, this difference is crucial to performance. Further description of architecture classes and their characteristics can be found in an earlier article.[1]

The remainder of this article

- reviews Valiant's argument[2] that the PRAM model is universal over tightly coupled and hypercube systems, but not over constant-valence-topology, loosely coupled systems — thus showing precisely how the PRAM model is too powerful to permit broad universality;
- discusses ways in which a model of computation can be restricted to become universal over less powerful architectures;
- introduces the Bird-Meertens formalism and shows how it is used to express computations in a compact way;
- shows the surprising result that the Bird-Meertens formalism is universal over all four architecture classes — the main result of the article — and shows that nontrivial restrictions of functional programming languages exist that can be efficiently executed on disparate architectures;
- discusses how the Bird-Meertens formalism is the basis for a programming language and shows that it is expressive enough to be used for general programming; and
- reviews other models and programming languages with architecture-independent properties.

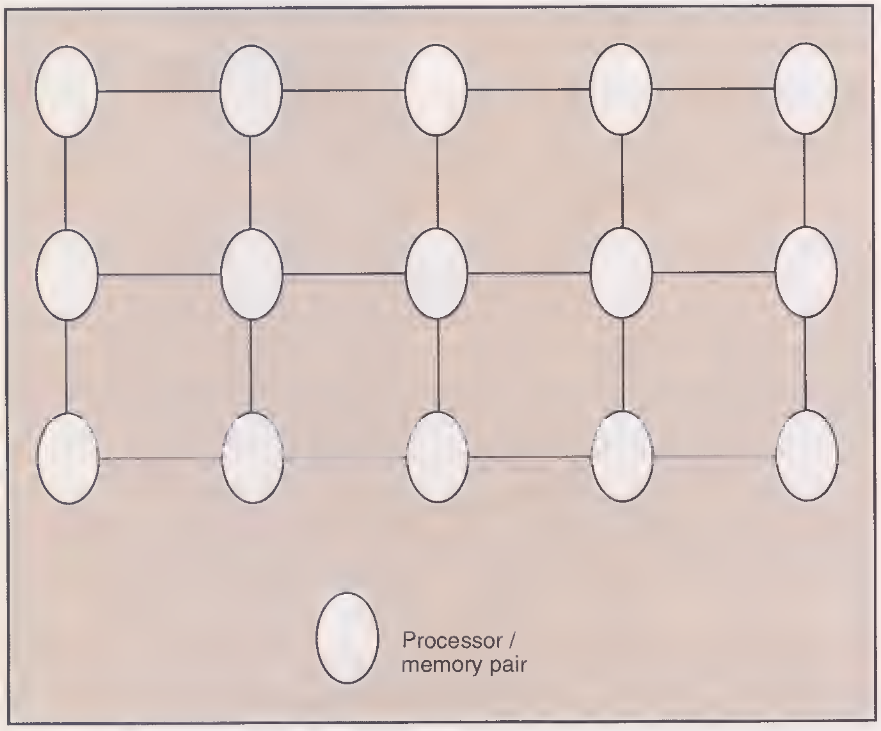


**Figure 4. A constant-valence MIMD computer.**

## Universal models

Valiant[2] carried out a careful analysis of the universality of the PRAM model over the four architecture classes described above.

The PRAM model is an abstract machine consisting of $p$ processors, each of which can, in unit time, carry out a local memory access, a global memory access, and a standard instruction. It is thus an approximation to a tightly coupled MIMD computer, but one that ignores the complications of memory and switch. The PRAM memory is considered to be a single, shared memory accessed through a zero latency switch (see Figure 5).

The sequence of steps executed by a single PRAM processor is called a thread. The number of time units a thread takes to execute is exactly the number of steps it contains. Because the only way a dependency between threads can be implemented is by one thread writing to memory and another reading the stored value, a dependency requires two-unit time steps. This is indicated by a two-unit arrow from one thread to another (see Figure 6).

A particular computation may be scheduled in many ways using different amounts of parallelism. Each schedule produces a trace consisting of threads. In what follows, we assume that the schedule chosen is as compact as possible, that is, it uses as little time and as few processors as possible. Thus, we assume, without loss of generality, that each thread contains a step at each time. For a computation of size $n$, we can characterize its parallelism and execution time by considering the width and length of its trace.

Suppose that the trace has $p(n)$ threads and $t(n)$ steps. By our assumption of compactness, the trace forms a $t(n)$ by $p(n)$ rectangle. We say that the cost of the PRAM computation is $t(n) \cdot p(n)$, representing the total amount of resources that must be used.

We define a computation model to be *universal* over an architecture class if there is a nontrivial architecture in that class that can emulate computations with time-parallelism products of the same order as their costs in the model. For example, the PRAM model would be universal over a particular architecture if a PRAM computation taking time $t(n)$ and $p(n)$ processors could be executed on that architecture in time $t(n)$ on $p(n)$ processors, or in time $2t(n)$ on $p(n)/2$ processors.

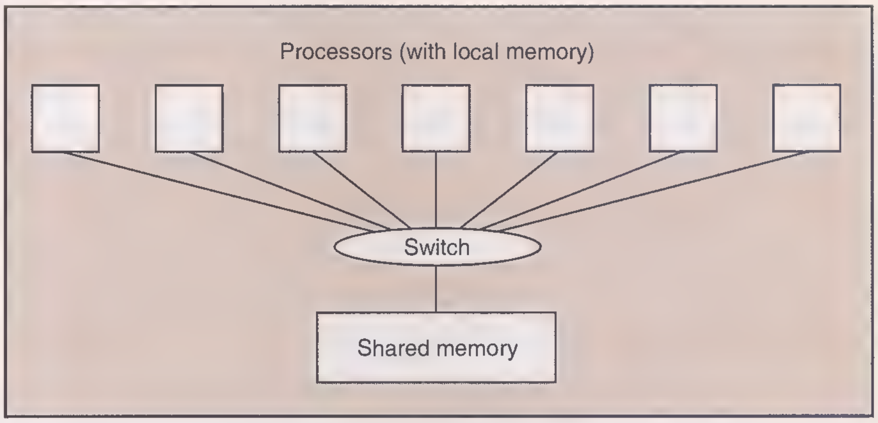


**Figure 5. A PRAM machine.**

**Emulation on tightly coupled computers.** Let us consider the PRAM model of computation implemented on a variety of architectures, beginning with the tightly coupled MIMD class. For this class, the switch is the performance bottleneck. Crossbar switches are very expensive in hardware, and optical switches are still highly experimental. Conventional dynamic switches require a traversal time logarithmic in the width of the switch, so that a $p$ processor system has an $\Omega(\log p)$ cost for each global memory access.

If we consider a straightforward implementation of this computation model on such an architecture (with $p(n)$ processors), we incur a time penalty, because each of the global accesses that takes unit time in the model takes logarithmic time on the real machine. Thus, the computation will take $t(n) \log p(n)$ time units, increasing the time-parallelism product by a factor of $\log p(n)$. Hence, this emulation is not universal. In fact, this example shows that we could not have defined a stronger form of universality in terms of execution time alone, since any real implementation incurs nonunit-time delays for references to distant data.

Fortunately, we can still construct a time-parallelism optimal simulation by reducing the number of processors to compensate for the increased memory latency. We reduce the number of processors to $p(n)/\log p(n)$ and use each processor to execute $\log p(n)$ threads of the computation in a kind of prescheduled multitasking. Each processor executes the first step of its first thread, then the first step of the second thread, and so on. After executing the first step of $\log p(n)$ threads, it executes the second step of the first thread, the second thread, and so on (see Figure 7). The elapsed time between successive steps of the same thread is $O(\log p(n))$ while the latency of the switch is

$$\log\left(\frac{p(n)}{\log p(n)}\right) < \log p(n)$$

Thus, ignoring potential contention in the switch, this emulation executes in time $t(n) \log p(n)$ using $p(n)/\log p(n)$ processors, giving an optimal time-parallelism product. This lower bound on switch traversal can be achieved by a result of Mehlhorn and Vishkin,[3] which shows that memory hashing can spread the memory references uniformly with high probability. This makes contention in the switch rare.

Notice that, to achieve this result, more parallelism must exist in the computation than in the machine, a property that Valiant calls *parallel slackness*.[4] The virtual parallelism of the computation must be much larger than the physical parallelism used to execute it. This really means that it doesn't help to use extra hardware for a computation — a thousand-way parallel algorithm can only make good use of a hundred-way parallel computer.

Unfortunately, this class of architectures does not seem to be a good candidate for long-term development. The problem lies in the scalability of the switch. Unless optical interconnects make a revolutionary

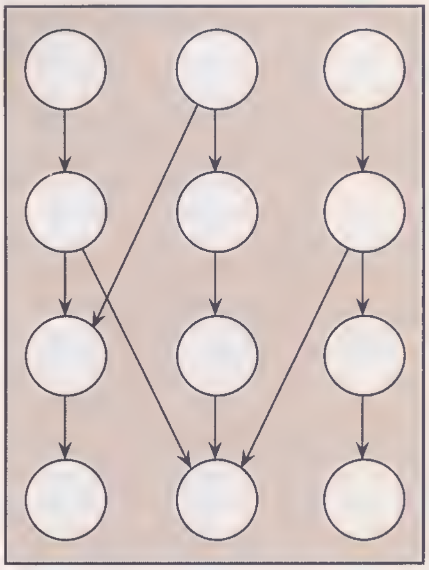


**Figure 6. Three threads showing dependencies.**

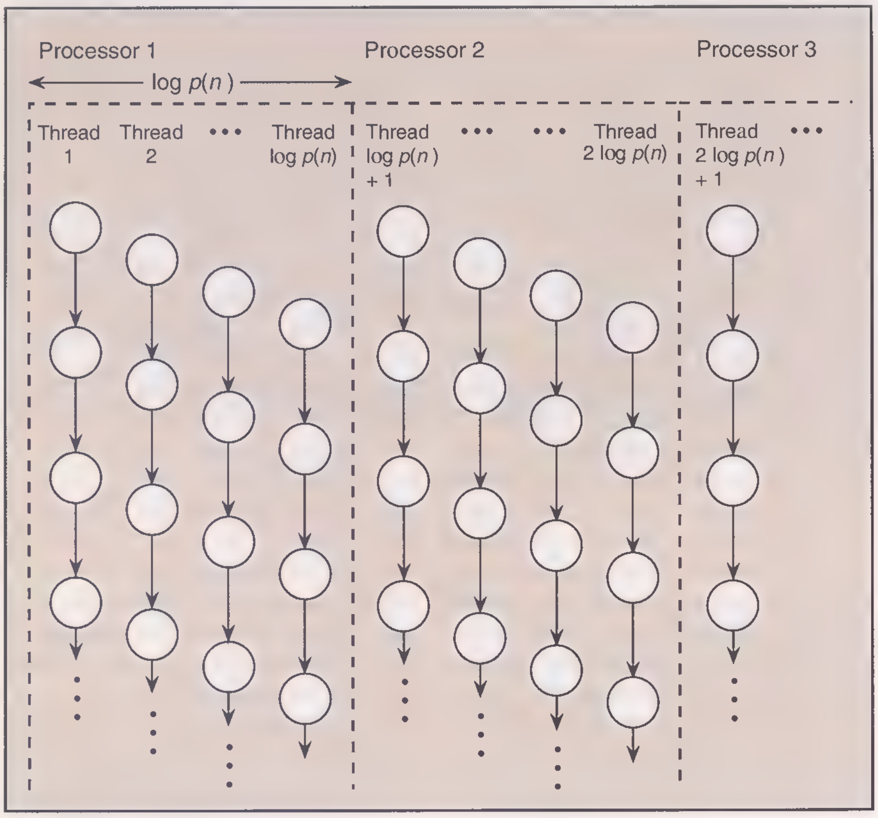


**Figure 7. Multitasking of threads onto physical processors.**

difference, today's estimate of the maximum possible switch size is about 10,000 × 10,000. As existing architectures are within an order of magnitude of this size, long-term prospects do not seem attractive.

**Emulation on hypercuboid computers.** Let us now consider emulating PRAM computations on a loosely coupled multiprocessor where the number of communication links from each processor grows as the logarithm of the number of processors. We call such systems *hypercuboid*, since the hypercube is the best known example of the class. Each processor has its own local memory, but can only access the data in the memories of remote processors by using a network of communication links.

The global references in our computation model must be transformed into messages to other processors that will not necessarily be adjacent. Hence, messages might travel through a number of intermediate communication links. We call the maximum number of links traversed the *effective diameter d*. By a theorem of Bokhari and Raza,[5] the diameter of any connected graph can be reduced to logarithmic by adding at most one new edge per vertex. There is thus little point in considering any static communication topology that has a diameter worse than logarithmic in the number of processors; such a topology could always be improved by adding only a single extra communication link to each processor. So, even a path topology could have its diameter reduced to logarithmic and its valence increased only to three. We therefore assume that in practical systems $d$ is bounded above by log $p$.

A direct emulation of the PRAM model will result in the same increase in execution time as for the tightly coupled implementation because unit-time global references take logarithmic time on the real machine. We use the same multitasking technique to reduce the number of processors, scheduling the steps just as before. The time between successive steps of the same thread is once again log $p(n)$, and thus enough time exists for references to the opposite extremity of a $p(n)$/log $p(n)$ processor machine to complete. The argument that this lower bound can be achieved in the presence of contention depends on two probabilistic results:

- memory hashing, to spread references uniformly; and
- two-phase randomized routing.[2]

Two-phase randomized routing channels a message from $A$ to $B$ by first sending it from $A$ to some randomly chosen processor $C$ by the straightforward shortest route, and then sending it from $C$ to $B$, again by the shortest route. The total distance travelled does not exceed twice the diameter of the network; this strange procedure reduces the probability of contention to an arbitrarily small amount. On the hypercube, it guarantees to deliver log $p(n)$ permutations in time log $p(n)$ with overwhelming probability. The total time taken for the emulation is $t(n)$ log $p(n)$ with parallelism $p(n)$/log $p(n)$, so that again we have a universal emulation. Parallel slackness is again required in the computation.

This architecture class is also problematic with respect to scalability. The number of links per processor depends on the size of computer in which it is embedded. Thus, it is not possible to build a scalable computer without replacing the processors whenever the system size is doubled.

**Emulation on constant-valence computers.** Let us now turn to loosely coupled, fixed-valence topology multiprocessors. In such computers, each processor has only a fixed number of communication links. Hence, such computers can be built as arbitrary size ensembles of the same basic processor. As before, we assume that the network diameter is at most logarithmic in the number of processors.[5]

A simple counting argument suffices to show that the restricted connection structure of such architectures must prevent universal emulation on them. Suppose we have a $p$ processor machine, that there are $\alpha$ communication links per processor, and that the effective communication diameter (that is, number of link traversals) of messages is $d$. In a single instruction step, as many as $pd$ message traversal requirements may be generated. Of course, these requirements are partly obligations for the future, as they must be satisfied on subsequent steps. The number of link slots available to transmit messages during a single step is $p\alpha$. Now $d$ is logarithmic in the number of processors while $\alpha$ is a small constant depending on the processor design. Therefore, on average, the communication system will not be able to deliver messages as quickly as they are generated. To compensate, the processors must be slowed by at least a factor of $d/\alpha$.

We use the same multitasking approach to schedule the steps of our computation model. Global references take time at least logarithmic in the number of processors extended by a factor of $d/\alpha$, giving a total execution time of

$$t(n) \cdot \log p(n) \cdot \frac{d}{\alpha}$$

on $p(n)/\log p(n)$ processors. The lower bound on communication time can be achieved for the cube-connected-cycles topology, using memory hashing and two-phase randomized routing. However, universal emulation is clearly not possible for this class of architectures.

The class of constant-valence topology MIMD computers is of great practical interest because computers in the class are scalable since the neighborhoods of each processor are homogeneous. Unlike the two classes previously considered, this class contains computers with extremely large numbers of processors.

**Emulation on SIMD computers.** Let us consider the fourth architecture class, the SIMD computers. Simulating our PRAM computation on a SIMD computer is difficult. Using a local table, a SIMD machine can simulate a MIMD machine by decoding each instruction broadcast into an instruction to execute. This broadcast instruction can be considered an index into the table; the table entries are then the instructions themselves. The question is whether such a simulation can be carried out without loss of universality. The general belief is that this cannot be so (it has the status of a folk theorem), but I have been unable to find an existing proof. The following theorem shows that universality cannot be maintained, except in the trivial case. It is based on the extra uniformity that a SIMD computer requires (a similar proof based on interprocessor communication is probably possible). The proof has the advantage that it applies even if the PRAM processors compute entirely independently.

*Theorem*: An arbitrary PRAM computation cannot be simulated on a SIMD architecture without increase in the time-parallelism product, except for the trivial simulation on a one-processor SIMD machine.

*Proof*: We assume that a SIMD architecture has a bounded bandwidth communication channel that broadcasts from the (single) instruction processor to the data processors. Call this bandwidth $x$. Let $IS$ be the cardinality of the instruction set of the data processors. Then $x$ is bounded above by $\log IS$.

Suppose the PRAM calculation uses $p$ processors. Then, there are $IS^p$ possible steps in the PRAM calculation. Suppose the PRAM calculation is simulated by a SIMD architecture with $m$ processors. Without loss of generality, we can assume that $m \le p$ since a PRAM computation that forces absolute dependencies between successive steps can always be constructed; such a program cannot make use of more than $p$ processors. Each step of the PRAM calculation is simulated by $p/m$ steps of the SIMD machine, with each set of $m$ operations chosen arbitrarily. There are $IS^m$ possible different configurations of $m$ of the PRAM operations, so that each SIMD step requires broadcasting $\log IS^m$ bits. With bandwidth $x$, this takes time

$$\frac{\log IS^m}{x}$$

Thus, the total time to simulate one step of the PRAM computation is

$$\log IS^m \frac{p}{mx}$$

and the time-parallelism product for the simulation is

$$\text{time–parallelism product} = \log IS \cdot \frac{mp}{x}$$

The time-parallelism product of the step of the PRAM computation is $p$.

Assuming that $x = \log IS$ (the usual way SIMD machines are designed), we see that the slowdown is the expected $m$; hence, when $m = 1$, the simulation remains universal, but it is suboptimal for all larger values.

The single step of the PRAM computation can be extended to an arbitrary number of steps; and a factor of $s$ appears in both of the calculations above. Hence, a full log $IS^m$ bits are needed to handle all possible $IS^m$ combinations. A PRAM computation long enough to use all possible combinations no matter how the $m$ are chosen can always be constructed.

This proof does not depend on any properties of the data processors used or any encoding of the instructions. It is based solely on the information flow across the instruction processor/data processor boundary. If the data processors are powerful enough to execute instructions stored as data, with direction from the instruction processor broadcasting fetch, decode, and execute, then the machine is best regarded as a MIMD computer and other arguments apply.

The class of SIMD architectures is also of long-term interest because computers in the class scale well; the only potential bottleneck is the fan-out from the instruction processor.

This shows that the PRAM model is universal over the classes of tightly coupled and hypercuboid multiprocessors. The PRAM model is not universal over constant-valence topology multiprocessors and SIMD computers. Unfortunately, this is as bad as it could be: Those architecture classes that can optimally emulate don't scale; those classes that do scale force a suboptimal emulation.

## Restricted computation models

The results in the previous section show why the PRAM model cannot be made universal over all four architecture classes. On the one hand, it requires frequent communication (possibly on every step); on the other hand, many diverse operations can take place simultaneously on different processors. The first creates problems in emulating the PRAM on communication-poor architectures; the second creates problems for SIMD architectures. A model universal over all four classes will have to be more restricted than the PRAM model, limiting communication richness and imposing more regularity on simultaneous steps.

A weaker model is not necessarily a bad thing. It was decided long ago that a sequential von Neumann machine's capability to treat instructions as data was not worth the problems it caused in software development and execution. Most von Neumann machines might as well be considered as Harvard architectures. Much the same reasoning was used to restrict imperative languages to a small set of control structures and to insist that programs be developed in a modular way.

There are many ways in which the PRAM model of computation might be weakened. There are two ways to treat the problems caused by communication: reduce the frequency of communication or reduce the distance each message travels. These suggest different, and largely incompatible, new models.

## Glossary

*Concurrent-read PRAM*: One in which simultaneous reads from the same memory location are allowed in unit time. It is usually implemented by replicating the value on its way from the memory to the processors.

*Concurrent-write PRAM*: One in which simultaneous writes to the same memory location are allowed in unit time. The actual value stored may vary: It may be a randomly chosen member of the set of values written, the result of applying an associative operation to the set of values, or the value from the lowest numbered processor participating.

*Cube connected cycles*: An interconnection topology that has many of the properties of the hypercube but only requires constant valence. Imagine a hypercube of dimension $d$. $d$ links converge on each corner. A cube connected cycles topology is obtained by removing the corners and replacing them with a cycle of size $d$.

*First-order functional programming*: Programming with functions that may take only data as their arguments. Many dataflow languages are first order.

*FP/FL*: Languages designed by Backus et al. They are built using first- and second-order functions. Much program transformation can be done using identities that are variable free; that is, they are identities of the functions only.

*KIDS*: Kestrel Interactive Development System, an algorithm development system built at the Kestrel Institute. It requires minimal user direction to derive algorithms of a number of common kinds: divide-and-conquer, dynamic programming, etc.

*Loosely coupled MIMD computer*: One in which each processor has its own local memory and is connected to other processor-memory pairs by an interconnection network. There is no direct access by one processor to another's memory.

*Memory hashing*: A technique for allocating variables to memory locations or modules such that, probabilistically, there is little chance of collisions during typical access patterns. It is based on uniform hashing functions.

*Model of computation*: An abstract but computable description of a computation. It usually corresponds to an abstract machine that can execute the model directly.

*OBJ*: A language based on order-sorted equational logic. Program code consists of equations that are interpreted as rewrite rules. Rewriting is done modulo commutativity and associativity and user-defined evaluation strategies may be defined. This provides a very flexible evaluation mechanism that can emulate other programming techniques.

*Parallel slackness*: The property of having much more virtual parallelism in a computation than is available on the physical machine executing it. Its presence often allows latency to be hidden.

*PRAM*: Parallel Random Access Machine, a popular computation model based on an abstract multiprocessor consisting of processors connected to a shared memory by a switch. In unit time, each processor can access its local memory or registers, access the shared memory, and perform a standard operation.

*Second-order functional programming*: The programming language contains (first-order) functions and data, but also functions that may take other functions as arguments. Usually the set of second-order functions is fixed.

*SIMD computer*: One in which a single instruction processor controls a set of data processors that simultaneously execute the operation broadcast by the instruction processor. The data processors are interconnected so that they can permute data among themselves.

*Tightly coupled MIMD computer*: One in which there is a large shared memory that is equally accessible to all processors. Because of the need to arbitrate the access to the shared memory, processors are not as independent as in loosely coupled MIMD computers.

*Two-phase randomized routing*: A routing algorithm that probabilistically reduces contention in static interconnection networks. It works by choosing a random destination for each message, routing it from the source processor to that random processor by a deterministic algorithm, and then routing it on to its destination using the same deterministic algorithm. This at most doubles the path length taken and spreads the load evenly across the communication paths.

*Universal*: A computation model is said to be universal over an architecture class if it can be simulated on that class without increasing the time-processors product required.

*VLIW*: Very long instruction word architectures use small fixed amounts of parallelism by constructing an instruction thread that contains a number of concurrent subthreads. This amount of parallelism can be extracted from ordinary sequential code at compile time using techniques rather like horizontal microcode compaction.

*XPRAM*: A variant of the PRAM suggested by Valiant. References to shared memory are counted as taking unit time but can only occur on every $L$th step in each processor.

Valiant has proposed the bulk synchronous parallel model, or XPRAM, a model which reduces the frequency of communication. It corresponds to a PRAM model in which nonlocal communication is constrained to occur no more frequently than every $L$ step in each processor. Choosing $L$ to be the size of $\log p(n)$ reduces the communication requirements enough to permit a universal emulation on the constant-valence topology architectures. *A fortiori*, universal emulations remain possible for tightly coupled and hypercuboid architectures.

The XPRAM can be regarded as a PRAM in which the granularity of the steps has increased to $L$. The instruction set of the XPRAM consists of all threads of length $L$ from the original PRAM. However, the structure of the computation now depends on the size of the target machine. Using a larger machine means choosing a larger $L$ and, hence, recasting the algorithm so that global communication is less frequent. This seems unsatisfactory and hard to implement. The kind of decisions required seem too difficult for a programmer, although they could conceivably be incorporated into a compiler. For example, work in compiling for VLIW (very long instruction word) architectures suggests that it is possible to recast algorithms for small values of $L$, but the technique will not work for arbitrary-sized $L$.[6] If $L$ is pragmatically bounded above by 10, computers are bounded to about a thousand processors, small even by today's standards. The memory allocation required is also quite unusual. There is a considerable advantage in locality of reference in a single processor, but memory hashing is required for interprocessor reference. Also, the XPRAM model does not address the problem of emulation on SIMD architectures.

The second way of restricting the computation model is to reduce the distance over which communication takes place. I call this *locality-based* computation. Under this model of computation, nonlocal references can only be to threads that are close under some metric, in fact within some constant distance. Such a computation model can be universal over constant-valence topology multiprocessors, because the effective diameter $d$ is a constant. As a result, it becomes possible to use only a small (constant) degree of multiplexing to hide latency — only constant parallel slackness is required. This allows more effective use of hardware.

First- and second-order functional programming are two attractive forms of locality-based computation. The appropriateness of first-order functional programming can be seen by regarding a computation as a dataflow graph consisting of nodes, representing functions, and arcs, describing the flow of data from one function to another. If the functions have only a small number of arguments and each produces only a single result, then the nodes are of low valence and can be mapped onto a constant-valence topology without "stretching" any of the arcs by more than a small amount. Hence, locality is always guaranteed.

Locality-based computation using second-order functions is even more attractive. The arguments to second-order functions must be functions that have some similarities. For our purposes, the similarities we are interested in are consistent communication patterns and uniform computation steps. If we can find a set of second-order functions such that each requires only constant locality, then the set can be used as a model of computation that will be universal over constant-valence topology multiprocessors and, therefore, over tightly coupled and hypercuboid multiprocessors as well. If the set of second-order functions has appropriate regularity, then we can consider emulations on SIMD computers as well.

The Bird-Meertens formalism[7] provides exactly such a set of second-order functions, although they were developed with rather different goals in mind. In the next section, I introduce this formalism, show that universal emulations over constant-valence and SIMD systems are possible, and discuss the expressiveness of the formalism.

## The Bird-Meertens formalism

The Bird-Meertens formalism consists of a set of theories built on a base algebra with unary and binary functions. Each theory captures the behavior of a particular class of data structures. The theory of lists has been well developed and some work has been done on the theories of trees and arrays.

A theory adds to the base algebra a set of second-order functions and laws that relate them. A program consists of a composition of functions, in much the same style as FP. The laws provide a set of meaning-preserving transformations that can be applied for optimization or regarded as rewrite rules in the style of OBJ.[8] The Bird-Meertens formalism thus owes something to APL and to the treatment of lists in conventional functional languages.

The theory of lists adds the following second-order functions to the base algebra: map ($*$), reduce ($/$), directed reduce ($\not\rightarrow$), accumulate ($\not\!\!\not\rightarrow$), prefix ($//$), filter ($\triangleleft$), *inits*, *tails*, and cross product ($\times$). If $f$ is a unary function, $\oplus$ a binary function written in infix notation, and lists are indicated by brackets, then we can define *map* applied to $f$ by

$$f * [a, b, c, ...] = [fa, fb, fc, ...]$$

The function *reduce* is defined by

$$\oplus/[a, b, c, ..., x] = a\oplus b\oplus c\oplus ...\oplus x$$

assuming that $\oplus$ is associative, so that bracketing is not needed on the right hand side. If $\oplus$ is not associative, we can define a directed reduce by

$$\oplus\not\rightarrow[a, b, c, ..., x] = (((a\oplus b)\oplus c) ...\oplus x)$$

The *accumulate* function defines a prefix computation over an operator that need not be associative. It is written $\not\!\!\not\rightarrow$ and is defined by

$$\oplus\not\!\!\not\rightarrow_e [a, b, c, ..., x] = [e, e\oplus a, (e\oplus a) \oplus b, ..., (...((e\oplus a)\oplus ...) \oplus x]$$

An associative version of *accumulate* called *prefix* can be defined by

$$\oplus// [a, b, c, ..., x] = [a, a\oplus b, ..., (...(a\oplus ...) \oplus x]$$

The *filter* operation provides selection. If $p$ is a Boolean predicate then

$$p \triangleleft [a, b, c, ..., x]$$

selects those elements of the list for which $p$ is true. Thus, its result is a (possibly empty) list.

The function *inits* computes the initial segments of a list and returns them as a list. Hence,

$$inits\ [a, b, c, ..., x] = [\ [\,]\ , [a], [a, b], ..., [a, b, c, ..., x]]$$

The function *tails* computes the final segments of a list, that is, *inits* of the reverse of a list.

The cross product operator forms the list of cross products of elements of two lists so that

$$[a, b, ..., m] \times_{\oplus} [n, o, ..., z] = [a\oplus n, a\oplus o, ..., a\oplus z, b\oplus n, ..., m\oplus z]$$

I illustrate the Bird-Meertens style with a simple example; many more examples, together with their derivations, can be found in references in the "Further reading" section. The *maximum segment sum* problem has been regularly discussed in programming literature. It can be stated as: Given a list of integers, find the contiguous sublist with maximum sum. Clearly, the following computation

$$mss = \uparrow/ \cdot +/* \cdot segs$$

where $\uparrow$ is the binary maximum operator, $+$ is integer addition, and *segs* computes all of the contiguous sublists of a list, meets this specification. The contiguous sublists of a list can be computed in the way implied by the following definition, in which $+\!\!+$ is the binary list catenation operator,

$$segs = +\!\!+/ \cdot tails* \cdot inits$$

This computation finds the maximum segment sum by computing all of the contiguous sublists of the given list, summing the elements of each, and then selecting the maximum of those sums. This is clearly a computationally expensive solution to the problem.

The laws of the theory of lists can be used to rewrite the solution; after a nontrivial derivation, the following solution results

$$mss = \uparrow/ \cdot \otimes \mathrel{/\!\!/\!\!\!\rightarrow}_0$$

where

$$a \otimes b = (a + b) \uparrow 0$$

The new version generates many fewer intermediate values and can be computed in two pipelined passes over the list. It uses the fact that, if the running sum falls below zero, then the corresponding sublist cannot be part of the maximum sum segment. The faster algorithm is not obvious, but is derivable by standard transformations. Some insight is required, but the process seems susceptible to automation. For example, the Kestrel Interactive Development System (KIDS)[9] can develop programs from specifications with minimal user input about the kind of algorithm that is appropriate — divide-and-conquer, dynamic programming, and so on.

## Demonstrating universality

To show that the second-order functions are universal over constant-valence topology multiprocessors, we must show that the time-parallelism product of an emulation is no worse (asymptotically) than the equivalent PRAM calculation, and that communication is sufficiently local that communication links are not saturated. In the following discussion, assume that lists are stored in a kind of normal form in contiguous processors, one element in each. Also assume that each list is of length $O(p)$ for a $p$-processor system, corresponding to our assumption that at least $p(n)$ processors were available to execute PRAM computations. Each processor is aware of the length of the list. In each of the following implementations, this information is sufficient to permit clean termination of each function.

Here, I give a case analysis for each of the second-order functions.

*Map*: The computation of a map requires each processor to apply a given function to the list element it holds. This can be done in constant time if we assume that the function is treated as part of the program, that is, it does not have to be broadcast at execution time. No interprocessor communication is required.

*Reduce*: The PRAM time complexity of reduce is logarithmic in the size of the list. I show that the same time complexity can be achieved using the following technique: A reduction can be carried out in a cube

**The Kestrel Interactive Development System can develop programs from specifications with minimal user input about the kind of algorithm that is appropriate — divide-and-conquer, dynamic programming, and so on.**

connected cycle network by first doing the reduction in each cycle. This takes time linear in the cycle length, that is, $\log p$. The remaining steps of the reduction can be done through dimension-by-dimension collapse. All of the values in one hyperplane are transmitted to the other parallel hyperplane, the $\oplus$ operation is performed at each corner, and the process is repeated in another dimension. Changing dimensions requires a shift around each of the cycles. All of the communication takes place with nearest neighbors in the topology.

*Directed reduce*: The PRAM complexity of a directed reduction is linear in the size of the list. It can clearly be implemented on the constant-valence multiprocessor in linear time, using only nearest neighbor communication, provided a Hamiltonian path exists. (A Hamiltonian path passes through every vertex in the graph exactly once. An example of a graph without such a path is a binary tree).

*Accumulate*: The accumulate function uses exactly the same communication pattern as the directed reduce, except that a copy of the partial result is left at each processor.

*Prefix*: The parallel implementation of prefix is due to Ladner and Fisher[10] and has a PRAM complexity that is logarithmic in the size of the list. In a constant-valence topology multiprocessor, their approach requires some nodes to transmit a logarithmic number of messages to others.

A constant locality parallel prefix can be computed in a cube-connected cycle topology as follows: First, compute the prefixes in each cycle of a cube connected cycles network. Then, use the following hypercube prefix algorithm. Each processor (with the result of a prefix from a cycle) holds two values, the sum of all the values in its current hypercube and its own partial sum (that is, the prefix value). Matching hypercubes are recursively merged into a larger dimension hypercube by having corresponding corners exchange their total sum, computing a new total sum, and those processors in the "upper" hypercube using the total sum from the other half to compute new partial sums. The hypercube part of the algorithm is clearly logarithmic, as is the initial prefix at each corner, for an overall logarithmic algorithm.

*Filter*: The filter operation returns a list in which those elements that do not satisfy the predicate have been removed. Its PRAM complexity is therefore logarithmic since, after elements have been deleted, those that remain must establish their new position in the list. On the constant-valence topology, the same operations must be done and take the same amount of time. However, one extra step, moving the values to contiguous processors, must also be done. The determination of new position can be calculated as

$$+ \mathrel{/\!\!/\!\!\!\rightarrow}_0 (\text{if } p \text{ then } K_1 \text{ else } K_0)$$

where $K_i$ is the constant $i$ function.

Moving the values to their correct positions may require arbitrary data movements. However, the destinations of each element of the list are unique, so the routing required is a permutation. Using two-phase randomized routing, an arbitrary permutation can be realized in logarithmic time in a network such as cube connected cycles. The use of two-phase randomized routing only occurs during filter operations. Each filter operation begins with a logarithmic time prefix operation; hence, repeated use of two-phase randomized routing is separated by a logarithmic time gap. This provides sufficient time to guarantee that successive routing steps will not interfere with each other. Thus, we can maintain an overall logarithmic time for the operation.

*Inits*: The PRAM complexity of inits is clearly linear since a linear number of processors must generate a quadratic amount of data. It can be computed on the constant-valence topology by circulating one copy of the list from left to right along a Hamiltonian path and adding the newly arrived element to the list being constructed at each processor. This takes linear time.

*Tails*: The tails operation can be computed in the same way as inits, except that the shift is from right to left.

*Cross product*: The cross product of two lists, each of which is in normal form, can be computed by circulating one list around a Hamiltonian cycle and forming products at each processor on each step. This takes time linear in the length of the lists and requires only nearest neighbor communication. The PRAM complexity of the operation is clearly linear.

Thus, the requirements for optimal evaluation on a constant-valence topology multiprocessor are the existence of a Hamiltonian cycle, the capability to do a tree-structured reduction in logarithmic time, and the capability to deliver an arbitrary permutation in logarithmic time. The second requirement is almost trivial because it amounts to requiring a log depth spanning tree of finite valence (which can simulate a binary spanning tree with no more than constant slowdown), and any interesting topology will have this property.

Both tightly coupled and hypercuboid multiprocessors are much less restricted than constant-valence topology architectures. Therefore, these results apply equally to those architecture classes.

The SIMD implementations of the second-order functions are no more complicated than the constant-valence topology implementations. In fact, the SIMD architecture is more powerful, since the requirement for locality is removed. It requires a certain regularity or uniformity in the computation because of the restriction that all processors must execute the same instruction at each step. The second-order functions we have been discussing all possess the required uniformity.

The details of the implementation of each of the functions on SIMD computers follow. We assume that the interprocessor topology is at least as rich as a constant-valence multiprocessor, so that communication patterns take the same amount of time as before. We need only show that processors are either executing the same operation or are idle during each step of the computation.

*Map*: As before, map can be applied in constant time and requires no communication.

*Reduce*: A reduction can be carried out by the obvious tree-structured algorithm in which half the processors participate for the first step, a quarter for the next step, and so on. The complete reduction is done after log $n$ steps.

*Directed reduce*: A directed reduction requires only a single processor to be active on each time step; hence, it takes the obvious linear amount of time.

**Knowing that the Bird-Meertens formalism is a computation model that is universal over four important classes of parallel architectures, it should also be clear that the Bird-Meertens formalism can be executed by uniprocessors and that many of its functions can exploit vector architectures.**

*Accumulate*: The accumulate operation is done in the same way as a directed reduction.

*Prefix*: The parallel prefix algorithm described above can be easily adapted to a SIMD architecture and still takes logarithmic time.

*Filter, inits, tails*: As before, these can be done using prefix computations.

*Cross product*: The same technique used for constant-valence computers can be used, giving a linear time algorithm.

I have shown that the Bird-Meertens formalism is universal over a diverse class of architectures, including computers that have limited communication or require uniformity of action. A computation model so restricted might not might have been expected to be powerful enough to be useful as a programming tool. The surprising result of this work is that such a universal model can be powerful enough to program most applications in a natural, although novel, way.

## Benefits of the Bird-Meertens theory

Knowing that the Bird-Meertens formalism is a computation model that is universal over four important classes of parallel architectures, it should also be clear that the Bird-Meertens formalism can be executed by uniprocessors and that many of its functions can exploit vector architectures. Thus, it is truly an architecture-independent programming language. It addresses many of the problems raised at the beginning of this article.

A natural question you might ask is whether the formalism is expressive enough or whether programmers will find it too restrictive. Making this case would require more space than is practical here, but interested readers can read the extensive variety of papers discussing applications. While this does require learning a new way of thinking about parallel computation, those who have used it seem to find no difficulty. In fact, several nonobvious improvements on existing algorithms have been discovered.

The choice of second-order functions is critical. Spivey has shown[11] how the basic set of second-order functions on lists arise as adjunctions between appropriate categories. Because such adjunctions are unique, all algebraic properties of list functions are captured by the laws that arise from the adjunctions. Thus, we can be sure that all algebraic properties of lists have been captured by these laws.

Software written in a language that is universal over a wide range of architectures is portable when it is written. But it also has a potentially long life span because it can be moved onto new hardware platforms as they are developed. Programmers also are less committed to a particular architecture class because they write in much the same way for any architecture.

The Bird-Meertens formalism also addresses some of the difficulties of software engineering for parallel architectures. In fact, it was developed with software engineering goals in mind, as an environment in which to do transformational program development. This approach postulates the ability to write down an initial solution that manifestly meets a given requirement as I did with the maximum segment sum problem. In fact, it may be little more than stating the requirements carefully. This solution can then be transformed, using the laws, which are actually algebraic identities, as meaning-preserving rewrite rules. This continues until a sufficiently efficient implementation is derived. Considerable experience with human derivation has been accumulated, and there seems to be no reason why the process could not be at least partially automated. I am investigating this possibility.

There is no guarantee that transformations will preserve the execution complexity of functions. It seems sensible to define new second-order functions whenever we can build implementations for them that are faster than the corresponding transformed functions would be. For example, a left accumulate can be defined in terms of a left reduction

$$\oplus \,/\!\!/\!\!\!\!\rightarrow_e = \otimes \,/\!\!\!\!\rightarrow_e$$

where

$$x \otimes a = x \mathbin{+\!\!+} [\text{last } x \oplus a]$$

but this definition is not computationally interesting since we know how to implement the left accumulate with the same parallel time complexity as the reduction. Obviously, this creates new problems since we cannot be certain that we know all useful transformation rules. Insight will be required to notice when some new composition admits a fast implementation. There are many interesting research problems here.

The analysis of the performance behavior of each of the second-order functions can be used to form the basis of a consistent complexity theory over different architectures. It can also be used to control program transformations, so that it is clear what an efficient solution actually is. Second-order functions such as reduce have recursion embedded within them, but it isn't visible to the programmer. However, when a cost measure is applied to functions such as reduce, the result is a first-order recursive function for which it is difficult to find general closed form solutions. We have shown that the complexity of the second-order functions is asymptotically the same as the PRAM versions, but the constants are almost always larger. Thus, it is interesting to consider architecture-directed optimizations that would help to reduce the practical disadvantages of the emulations. Many interesting research questions remain.

## Other architecture-independent languages

Attempts to find architecture-independent computation models or programming languages take two approaches: restricting the PRAM model by requiring the use of new primitives; or removing restrictions such as scheduling from the PRAM model.

Models that restrict the PRAM are easier to implement because their demands on the underlying hardware are more predictable. Thus, providing performance guarantees for such models is usually possible. At first glance, it seems paradoxical that a model is restricted by adding primitives, but performance gains are achieved by restricting programmers to using the new primitives rather than all of the flexibility of the original model. For example, adding a structured *if* and *while* to an imperative language is a restriction of general programming, even though it involves adding new language constructs.

The first restriction I will consider is the addition of concurrent read or concurrent write at the same memory location. These can be regarded as parallel read or write primitives. As with all such primitives, these are added because they can be efficiently implemented. On a tightly coupled MIMD computer, the paths through the switch from each processor to a fixed memory location form a tree. If the internal nodes of this tree can do simple computations, they can merge requests for reads from the same location and then distribute the data when it returns from memory. Writes to the same location can be merged on their way through the switch using any of a number of rules: keep one value and discard the others, apply an associative operation to the two values and transmit the result, and so on. Thus, with the addition of suitable (quite expensive) hardware, concurrent read and write can be implemented at the same time cost as ordinary memory access that is logarithmic in the number of processors.

Another primitive that can be added to the PRAM model is the *scan*, suggested by G. Blelloch (see "Further reading"). A scan is essentially a parallel prefix. Again, in a tightly coupled MIMD computer, this operation can be implemented by the switch in time logarithmic in the number of processors.

A.G. Ranade has suggested a further restriction of the PRAM model (see "Further reading"). He describes a new primitive called *multiprefix* that generalizes *scan*. Suppose that some set of $k$ processors references a variable $A$, the processors are ordered, and $\oplus$ is a binary operation. If the initial value of $A$ is $a$, then the execution of the multiprefix $MP(A, v_i, \oplus)$ by processor $i$ results in it acquiring the value $a \oplus v_1 \oplus v_2 \oplus \ldots \oplus v_i$ and the variable $A$ ends up with the value $a \oplus v_1 \oplus \ldots \oplus v_k$. Ranade shows that the multiprefix operation will terminate (with overwhelming probability) in $\log p(n)$ steps. Thus, on a tightly coupled MIMD computer, it is no more expensive in time than memory reference. The Bird-Meertens formalism can be regarded as an extension of this approach to a set of primitives that is in some sense complete while still being efficiently implementable.

The second approach is to build models stronger than the PRAM in which the programmer must say less about scheduling and communication. Such models are harder to implement because the programmer provides less information. But, of course, the programmer's job is easier for the same reason.

Macro-dataflow is a popular model of this kind. It is stronger than the PRAM model because the programmer no longer needs to specify the order of execution of parts of the program, and there is no longer any explicit memory. Instead, scheduling is inferred at runtime by the presence of arguments ("firing rule"), and memory is replaced by tokens traversing arcs. Dataflow machines attempt to create execution schedules dynamically as a program executes. Doing this efficiently is the major challenge of this approach.

Another programming language, or perhaps abstract machine model, is W.J. Dally's parallel machine interface (see "Further reading"). The PMI consists of a set of mechanisms that can be efficiently supported by the underlying parallel computers while being rich enough to allow programming in a number of suitable styles. The mechanisms act as an abstract machine for which higher software layers can be targeted, decoupling the software development process from the underlying implementation.

The abstract machine is considered to consist of a number of nodes, each of which can execute tasks. Tasks have their own local memories, called segments. Segments contain the entire context of a task, including a flag indicating whether they are available to run, and memory locations are tagged with their full/empty status. Tasks on the same node can access the segments of other tasks.

Communication and synchronization are both implemented by a single *send* primitive that transfers a block of data to another node, places it in a segment, and flags the segment as ready to execute. Thus, messages can trigger actions on remote nodes. These actions can mimic typical *receive* operations, and they can also suspend or resume other tasks on the destination node.

The PMI approach cannot provide any guarantees about the performance of a computation on a range of implementation architectures. However, it is possible to compute the complexity of a computation on each individual architecture by examining the implemented cost of the primitive.

Parallel languages that require even less of the programmer have also been suggested. One of the best known is Linda, which is perhaps best considered as a memory abstraction that may be used in many programming languages. Thus, C-Linda, Ada-Linda, and so on can all be built on the same abstraction. Linda provides the abstraction of a shared, content-addressable memory that can be accessed by any process with equal ease. Both scheduling and communication are handled by the system. Programmers need only specify dependencies. The memory is called tuple space, and the entities it contains are called tuples.

Four access mechanisms are provided:

- *in* selects, removes, and returns a tuple from the tuple space based on the number, types, and supplied values provided in the call; missing values are filled in from the values in the tuple;
- the primitive *read* selects and copies a tuple from the tuple space (so that the tuple is left in the tuple space for subsequent accesses);
- the primitive *out* places a tuple in the tuple space; and
- the primitive *eval* gets a tuple from the tuple space, treats it as executable code, and schedules it for execution.

This abstraction is very powerful because of its capability to access data based on partial descriptions and because of the capability to start new tasks with little overhead. Storage, synchronization, and communication are all managed by the same mechanism. In addition, tuple space might survive the execution of a single program; that is, it has some of the characteristics of a file.

However, the power of the abstraction creates problems for the implementer, particularly on a loosely coupled architecture. If each tuple exists only once in the tuple space, then accesses to it from other processors are necessarily slow; if the tuple is replicated, then it is hard to implement the semantics of *in* because *in* must guarantee that all copies other than the one returned are destroyed. The simulation of a large-content addressable memory, particularly one that may be accessed by a variety of different key patterns, is also challenging and creates substantial overhead. Thus, although Linda provides a very pleasant environment for the programmer, and one that makes portability straightforward, it does so at the expense of any guarantee about performance.

Linda also makes it difficult to capture fine-grained parallelism. For instance, the obvious implementation of *+/a* by

```
in(a, x)
in(a, y)
out(a, x+y)
```

fails because of the potential deadlock when the number of processors is as large or larger than the number of elements of the list. This seems unsatisfactory.

Another approach that requires no explicit control of scheduling or communication is K.M. Chandy and J. Misra's Unity and its relatives (see "Further reading"). A Unity program consists of a loop around a block of guarded statements. On any iteration of the loop, a statement whose guard evaluates to *true* is executed. The choice of statement is made nondeterministically. Such a program can be executed by a parallel computer by executing all statements whose guards are true in parallel. This simulates multiple iterations of the loop. Unity was developed as a language for reasoning about computation rather than executing computation. Whether Unity is implementable at any practical cost is not clear.

However, a related approach called *action systems* has been considered as an executable language. For example, R.J.R. Back shows how a sequential program in the formalism can be refined to a parallel program suitable for either shared-memory or loosely coupled architectures (see "Further reading"). The refinement preserves total correctness, so that the attractive reasoning properties can be retained even while operational notions such as parallelism are integrated. An implementation on a transputer system with reasonable performance exists.

The somewhat surprising result that a nontrivial computation model exists that is universal over four major classes of parallel architectures (tightly coupled, SIMD, hypercuboid, and constant-valence topology multiprocessors) provides the basis for an architecture-independent programming language. Programs developed in this language can be moved from machines in one architectural class to machines of another class without reprogramming and without paying performance penalties of more than constant factors. Thus, this new model provides the flexibility of Linda, but provides stronger guarantees about performance. Stronger results may yet be obtained, depending on progress in exploiting program optimization. This question is being investigated.

The programming language for this new model is closely related to the Bird-Meertens formalism. This relationship makes it possible to demonstrate that the language is quite expressive, since extensive algorithm development has been done within that formalism. Categorical results also increase confidence that the limited set of second-order functions included in the language is rich enough to capture all properties of general lists. In addition, these results provide a large set of algebraic identities that can be used for optimization. ■

## Acknowledgments

Discussions with Bill McColl, Gaétan Hains, Laurie Hendren, and Kieran Herley greatly helped me with the content and presentation of this article, and I am grateful to them for their help. I also thank members of the Programming Research Group at the University of Oxford in England for their hospitality during my stay as a visiting researcher. The Natural Sciences and Engineering Research Council of Canada supported this work.

## References

1. D.B. Skillicorn, "A Taxonomy for Computer Architectures," *Computer*, Vol. 21, No. 11, Nov. 1988, pp. 46-57.
2. L.G. Valiant, "General-Purpose Parallel Architecture," Tech. Report TR-07-89, Computer Science Dept., Harvard Univ., 1989.
3. K. Mehlhorn and U. Vishkin, "Randomized and Deterministic Simulation of PRAMs by Parallel Machines with Restricted Granularity of Parallel Memories," *Acta Informatica*, Vol. 21, 1984, pp. 339-374.
4. L.G. Valiant, "Optimally Universal Parallel Computers," *Proc. Royal Soc.*, 1987.
5. S.H. Bokhari and A.D. Raza, "Augmenting Computer Networks," in *Proc. Int'l Conf. Parallel Processing*, IEEE Computer Soc., Aug. 1984, pp. 338-345.
6. A. Aiken and A. Nicolau, "Optimal Loop Parallelization," in *Proc. SIGPlan 88: ACM Conf. on Programming Language Design and Implementation*, R. Wexelblat, ed., 1988, pp. 308-317.
7. R.S. Bird, "Algebraic Identities for Program Calculation," *The Computer J.*, Vol. 32, No. 2, Feb. 1989, pp. 122-126.
8. J.A. Goguen and T. Winkler, "Introducing OBJ3," Tech. Report SRI-CSL-88-9, Computer Science Lab., SRI Int'l, Aug. 1988.
9. D.R. Smith and M.R. Lowry, "Algorithm Theories and Design Tactics," in *Math. Program Construction*, Springer-Verlag

Lecture Notes in Computer Science 375, June 1989, pp. 379-398.

10. R.E. Ladner and M.J. Fisher, "Parallel Prefix Computation," *J. ACM*, Vol. 27, 1980, pp. 831-838.

11. J.M. Spivey, "A Categorical Approach to Theory of Lists," in *Math. Program Construction*, Springer-Verlag Lecture Notes in Computer Science 373, June 1989, pp. 399-408.

## Further reading

Ahuja, S., et al., "Matching Languages and Hardware for Parallel Computation in the Linda Machine," *IEEE Trans. Computers*, Vol. 37, No.8, Aug. 1988, pp. 921-929.

Back, R.J.R., "A Method for Refining Atomicity in Parallel Algorithms," in *Parle 89, Parallel Architectures and Languages Europe*, Springer Lecture Notes in Computer Science 366, June 1989, pp. 199-216.

Back, R.J.R., and K. Sere, "Stepwise Refinement of Action Systems," in *Math. Program Construction*, Springer Lecture Notes in Computer Science 375, June 1989, pp. 115-138.

Backus, J., "Can Programming be Liberated from the von Neumann Style: A Functional Style and Its Algebra of Programs," *Comm. ACM*, Vol. 21, No. 8, Aug. 1978, pp. 613-641.

Backus, J., et al., "FL Language Manual, Parts 1 and 2," Tech. Report RJ7100, IBM Almaden Research Center, Oct. 1989.

Bird, R.S., "A Calculus of Functions for Program Derivation," Oxford Univ. Programming Research Group Monograph PRG-64, 1987.

Bird, R.S., "An Introduction to the Theory of Lists," in *Logic of Programming and Calculi of Discrete Design*, M. Broy, ed., Springer-Verlag, 1987 pp. 3-42.

Bird, R.S., "Lectures on Constructive Functional Programming," Oxford Univ. Programming Research Group Monograph PRG-69, 1988.

Blelloch, G., "Scans as Primitive Parallel Operations," in *Proc. Int'l Conf. Parallel Processing*, CS Press, Los Alamitos, Calif., Order No. 783, Aug. 1987, pp. 355-362.

Chandy, K.M., and J. Misra, *Parallel Program Design: A Foundation*, Addison-Wesley, 1988.

Dally, W.J., "Universal Mechanisms for Concurrency," in *Parle 89, Parallel Architectures and Languages Europe*, Springer-Verlag Lecture Notes in Computer Science 365, June 1989, pp. 19-33.

Meertens, L.G.L.T., "Algorithmics — Towards Programming as a Mathematical Activity," in *Proc. Dutch Center for Math. and Computer Science (CWI) Symp. Math. and Computer Science*, North-Holland, 1986, pp. 289-334.

Ranade, A.G., Fluent Parallel Computation, PhD thesis, Yale Univ., 1989.

Valiant, L., "A Bridging Model for Parallel Computation," *Comm. ACM*, Aug. 1990, pp. 103-111.

**David B. Skillicorn** is an associate professor in the Department of Computing and Information Science at Queen's University, Kingston, Ontario, Canada. His research interests are in parallelism, spanning architectures, languages, and compilers.

Skillicorn received a BSc in 1978 from the University of Sydney, Australia, and a PhD in 1981 from the University of Manitoba, Canada. He is a member of the IEEE Computer Society and the ACM.

Readers can write to Skillicorn at the Department of Computing and Information Science, Queen's University, Goodwin Hall, Kingston, Ontario, Canada K7L 3N6, e-mail skill@qucis.queensu.ca.

**13TH INTERNATIONAL CONFERENCE ON**

**SOFTWARE ENGINEERING**

**SYSTEM DESIGN: RESEARCH & PRACTICE**

**Stouffer Hotel ■ Austin, Texas ■ May 13-16, 1991**

**Information on Technical Programs, Tutorials, Workshops, and Other Activities:**

Barbara Smith • MCC Software Technology Program
P.O. Box 200195 • Austin, TX 78720 • 512/338-3336 • FAX 512/338-3899 • basmith@mcc.com

**For Tools Fair Information:**

Laurie or John Werth • Department of Computer Science
University of Texas at Austin • Austin, Texas 78712 USA • 512/471-7316 • FAX: 512/471-8885
lwerth@cs.utexas.edu • jwerth@cs.utexas.edu

acm Sponsored by ACM Special Interest Group on Software Engineering IEEE Computer Society Technical Committee on Software

IEEE THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

## CALL FOR PARTICIPATION

# 1991 INTERNATIONAL CONFERENCE ON COMPUTER PROCESSING OF CHINESE AND ORIENTAL LANGUAGES

**August 13–16, 1991**

**Taipei, Taiwan**

**Sponsored by Chinese Language Computer Society**

**National Tsing Hua University, Hsinchu**

**Academic Sinica, Taipei**

### SCOPE

This conference serves as an international forum for researchers, system developers and users of information systems which process Chinese, Japanese, Korean and other oriental languages. Papers are invited for, but are not limited to, the following subject categories.

- Intelligent terminals and Workstations
- Character recognition
- Font design and generation
- Speech recognition and synthesis
- Coding scheme
- Database and information system design
- Expert systems and applications
- Man–machine interface

### SUBMISSION OF PAPERS

Three copies of full paper written in English not longer than 15 double spaced pages should be sent to either

Prof. Wen–Hsing Hsu
Dept. of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan, 30043

Prof. Yaohan Chu
Department of Computer Science
University of Maryland
College Park, MD 20742

- Paper submission deadline: January 15, 1991
- Notification of acceptance: April 15, 1991
- Final Manuscript due: May 31, 1991

### TUTORIALS

Proposals for tutorials should be submitted by April 1, 1991 to:
Prof. Ching C. Hsieh
Computing Center
Academic Sinica
Nankang, Taipei, 11529

### EXHIBITIONS

Systems of novel design and advanced technology are invited for exhibition. Please contact
Prof. Shi–Nine Yang
Dept of Computer Sciences
National Tsing Hua University
Hsinchu, Taiwan 30043

### CONFERENCE CO–CHAIRPERSONS

Dr. Chao–Ning Liu
IBM Corporation
Yorktown Heights, NY

Prof. Hsiao–Chuan Wang
National Tsing–Hua University
Hsinchu, Taiwan

### PROGRAM CO–CHAIRPERSONS

Prof. Yaohan Chu
University of Maryland
College Park, MD

Prof. Wen–Hsing Hsu
National Tsing–Hua University
Hsinchu, Taiwan

### TUTORIALS COORDINATOR

Prof. Ching C. Hsieh
Academic Sinica
Taipei, Taiwan

### LOCAL ARRANGEMENT CHAIRMAN

Prof. Shi–Nine Yang
National Tsing–Hua University
Hsinchu, Taiwan

### PROGRAM COMMITTEE

Lydia Chan (Singapore)
Keh–Jiann Chen (Taipei, Taiwan)
Yaohan Chu (Maryland)
Wen Hsing Hsu (Hsinchu, Taiwan)
Jyh–Sheng Ke (Taipei, Taiwan)
C. N. Liu (New York)
Wen–Hsiang Tsai (Hsinchu, Taiwan)
Hisao Yamada (Tokyo)
Shi–Kuo Chang (Pittsburgh, Penn)
Francis Chin (Hongkong)
Ching C. Hsieh (Taipei, Taiwan)
Jack K. T. Huang (Taipei, Taiwan)
Lin–Shan Lee (Taipei, Taiwan)
Ching Y. Suen (Montreal)
Hsiao C. Wang (Hsinchu, Taiwan)
Shi–Nine Yang (Hsinchu, Taiwan)

# Philosophies for Engineering Computer-Based Systems

Harold W. Lawson
Lawson Publishing and Consulting, Inc.

In recent years, attention has focused on software engineering — especially its computer-aided aspects— as a "cure" for the life-cycle problems of complex computer-based systems. Unfortunately, this emphasis is like placing the cart before the horse. Software engineering methods and tools are important, but they should be the natural result of a well-developed philosophy for solving the application problem.

By philosophy I mean a unifying common view of how a problem or class of problems shall "in principle" be treated. The view, which is based on concepts, must be commonly held by project team members and all other parties with vested interests. It involves the development of a strategy from which decisions (large and small) emanate. The philosophy should be documented, most often via appropriate paradigms and models, for communication to others. Once the philosophy is understood and practiced, the decisions will follow a common pattern. The philosophy "way of doing business" leads to both explicit and implicit knowledge. With a common view, details for solving large and small problems will be convergent instead of divergent, because there are no diverse views or misunderstandings. A good common philosophy contributes to an esprit de corps within project teams and organizations, motivating people to follow and propagate the philosophy.

**A sound problem-relevant philosophy is the key to achieving successful implementation of complex computer-based systems. Software engineering methods and tools will naturally flow from this foundation.**

A dictionary definition of the word philosophy appropriately conveys several important points made in this article:

> philosophy. 1. the rational investigation of being, knowledge, and right conduct. 2. a system or school of thought: the philosophy of Descartes. 3. the basic principles of a discipline: the philosophy of law. 4. any system of belief, values, or tenets. 5. a personal outlook or viewpoint. 6. serenity of temper.
>
> Source: *The Collins Dictionary and Thesaurus*, William Collins Sons & Co. Ltd., 1987.

Unfortunately, many projects start without a philosophy. Those who have initiated or participated in projects guided by a clear philosophy know exactly what I mean. Those who have not have missed a vital professional experience. I hope this article will enlighten the latter group.

A philosophy comes from within the organization, often based on stimulus from one or a very few creative persons who are what Brooks[1] referred to as "the great designers." (Sometimes this includes an external person with broad experience who provides the starting point). Over time, the philosophy achieves common acceptance. In contrast, when methods and tools (in the absence of an internal problem-relevant philosophy) are "dictated" from outside the project organization, project members often resist and adjust only reluctantly.

## The case of Simula

To illustrate the importance of a philosophy, let's consider the development of Simula. This programming language[2] was designed in the early 1960s as a result of a philosophy for simulating industrial work environments and social systems. Dahl, Myhrhaug, and Nygaard, the Simula founders, later assisted Norwegian labor unions in gaining insight into various produetion models. They evolved concepts they felt were important for programming industrial work environment simulators, including the notions of classes, subclasses, objects with attributes, and inheritance. These concepts became the heart of the Simula philosophy, which lives on among avid Simula users and is perpetuated by the Simula User's Group.

Dahl, Myhrhaug, and Nygaard used Algol-60 as a base ("carrier") for their concepts. Thus, Simula was born as a further development of Algol. At approximately the same time, I developed the list processing facilities of the PL/I programming language.[3] While these facilities can be used, via detailed programming, to accomplish goals similar to those of Simula, the background for their development was quite different. In contrast to the specific problem-related orientation of Simula, PL/I was developed as a joint effort of many interests. It reflects rather pragmatic decisions by parties with diverse interests and views. PL/I, as a large committee effort, suffered greatly from the lack of a commonly agreed upon philosophy. On the other hand, the list processing facilities (an add-on to the original language specifications) were based on the concepts of based and pointer variables. These concepts have proved to be a widely accepted philosophy in their own right. They have been widely used in PL/I programs[4] and "carried" forward into many PL/I dialects and subsets. Also, they provided the backbone of the C programming language.

Simula was based on a philosophy aimed at solving a particular class of problems. It was good at doing that job; however, it also became useful for a wider class of programming problems. Its problem-related philosophy-first basis led to a successful solution. The abstract notions developed by the Simula pioneers have finally resulted in an emphasis on the importance of object-oriented design, as well as various concrete carriers of the concepts, for example, the programming languages Smalltalk, C++, Objective C, and Eiffel.

## "In vogue" philosophies

Some philosophies attract a great deal of attention in the computing community, becoming contemporary fads. Although it took a long time, the Simula philosophy is one example of an "in vogue" philosophy. Other examples are the Unix philosophy centered on processes and pipes, and the philosophy related to windows and the desktop paradigm. Both philosophies have been implemented through a variety of concrete carriers. In addition to the object-oriented philosophy started by Simula, these faddish philosophies are currently guiding a significant portion of our view of modern computer-based systems development.

## Philosophy decay

Philosophies, once established, must be nurtured and treated with respect; otherwise, they deteriorate. One of the most significant (and costly) examples of philosophy decay was the development of the OS/360 operating system during the early 1960s. The first six months of project planning resulted in a well-thought-out and neatly documented operating system philosophy for the new family of computers. There was one design notebook. Then came the marketing demands to incorporate a variety of features, special requirements for related project groups, pressing time schedules, and other demands. That resulted in an explosion of the project documentation and the eventual involvement of a cast of thousands around the globe. The decay was rapid and significant and has radically influenced IBM, its customers, consultants, and many others since that time. The OS/360 experience with philosophy decay is extreme; however, it has been observed in many other large computer projects.

## The pragmatic gap and "feature-itis"

The period between the mid-1960s and the mid-1980s — the "pragmatic gap" — was a period of both great confusion and economic opportunity in computing. It was also a period filled with "feature-itis"; that is, more was better. Commercial products were filled with features that clearly reflected their companies' lack of unifying central philosophies and the prevalence of pragmatism.

The three philosophies that are in vogue — object orientation, Unix, and windows — should improve our product understanding and our ability to effectively use computer systems technology. However, although one would hope that these philosophies would not deteriorate and suffer from "feature-itis" as did their predecessors, there are already indications of philosophy decay. Consider, for example, the major extensions that have been added to Unix for process synchronization, interrupts, and priority. These extension have led some experts to reexamine the situation and create a new philosophy (the Mach operating system). Another example is the tendency towards feature-itis window systems, such as Motif.

## Philosophy versus software engineering

Some may argue that a sound project philosophy is a part of good software engineering design. If so, the philosophy must be the abstract set of concepts guiding all further developments.

In many cases, software engineering refines existing practice by reacting to concrete, often pragmatic problems in the "management" of the software life cycle. Philosophies are based on principles, concepts, and strategies, whereas software engineering is based on methods, mechanisms, and tools. This is not to say that the methods, mechanisms, and tools are not important, nor that they lack philosophy, since they are normally built according to philosophies. As long as the philosophies of these methods, mechanisms, and tools are congruent with the problem philosophy, all is well. Unfortunately, software engineering often means solving the "imagined" (frequently pragmatic) problem and not the actual system realization problem.

Another aspect of software engineering common to faddish computing is "hopping on the bandwagon." Those who join the bandwagon expect the methods and tools to provide salvation without understanding the source of their actual problems. Overstructuring the problem caused by some computer-aided software engineering (CASE) methods and tools can lead to overmanagement and overspecification. This in turn leads to volumes of human and/or computer-generated documentation that nobody wants or bothers to read.

In the next four sections of this article, I will discuss several philosophy-relevant factors and illustrate experiences where philosophies have played a central role in realizing complex real-time applications.

## Contextual aspects

Many factors relate to both the development of a philosophy and the absence of one. They reflect principles, points of view, and human psychology, as well as national traditions, organizational structures, cultures, practices, and decision making.

**Problem versus computer system fixation.** The first, most basic issue is the question of fixation. While fixation is found in all computer-based system projects, we'll focus on real-time systems to illustrate the problem.

Computer systems were first widely used in real-time applications in the mid-1970s when the microprocessor and related components were introduced. Many engineers who implemented real-time systems with analog technology resisted using computer-based solutions. In fact, some speculated that this role of the computer was a passing fad.

We now have a history of computer-based, real-time system solutions. Many computer experts solve real-time problems, frequently without the problem-relevant insight of the engineer. Narrowly focusing on either the problem's engineering aspects or its computer system solutions leads to uncertainty and communication difficulties, thus hindering appropriate philosophy building. No group of engineers, computer experts, or mixture thereof can develop an appropriate philosophy by sticking to its own narrow thinking.

We are all products of our experiences, as broad or as narrow as they may be. The majority of computer experts view problem solutions in terms of specific artifacts of their profession, for example,

**The fixation on computer-related artifacts by those responsible for computer-based systems has hindered proper development.**

- specific computer systems,
- languages,
- operating systems,
- shells,
- methods, and
- tools.

These artifacts become the media through which they think about and see the problem. They tend to "immediately" cast the problem in terms of one or more of the artifacts. In the real-time context, they may also see the problem in other computer-related terms, such as

- interrupts,
- communication,
- synchronization,
- rendezvous,
- scheduling, and
- state models.

The computer experts a priori view often hides the real problems of developing a sound, problem-relevant common philosophy. On the other hand, a broad knowledge and understanding of the possibilities and limitations of the artifacts is an important ingredient in building a relevant project philosophy. The essential point is not to let dominant orthogonal artifacts limit the problem solutions or tie the problem to inappropriate approaches.

**Natural vs. forced solutions.** A philosophy that is natural for the solution of a problem or class of problems will lead to paradigms and models, followed by methods, mechanisms, and tools that are consistent with the philosophy. It will clearly lead to other appropriate computer-related artifacts. (Three specific philosophy examples representing this form of project development in Europe are presented in this article.) Following this path of setting the horse before the cart (that is, philosophy before method and tools) leads to a much more rational, self-perpetuating approach to solving the life-cycle problems of complex computer-based systems. This fact is confirmed in a study of several Swedish computer-based system projects.[5] The study also confirms that the success of the project can be traced to one or a few great designers who have guided the philosophy and received full management support.

When the problem is viewed in terms of specific artifacts, the view brings a priori conditions of what can and cannot be accomplished. Thus, from the beginning, a number of degrees of freedom are removed, restricting the solution domain. In this environment, the problem solution is often forced into the framework of computer-related artifacts that conflict with the natural problem philosophy. Eventually, this results in a variety of long-lasting and costly practical problems, including negative attitudes among the project team members.

The fixation on computer-related artifacts by those responsible for computer-based systems has hindered proper development. In the relationship

Problem <—> Computer System

the emphasis has been on the artifacts' capabilities. In fact, it is the problem that is important; thus, the artifacts should be applied to the problem only when they are consistent with the philosophy. When proposed methods and tools are orthogonal to the natural solutions of the problem, new philosophy-related methods and tools must be developed.

To illustrate natural versus forced solutions, let's consider a concrete example: using Ada to develop software for problems where the natural solution calls for cyclic execution philosophy, as described by Baker and Shaw.[6] In their article, Baker and Shaw illustrate that by various unnatural means, Ada — which is based on the rendezvous synchronization philosophy — can implement cyclic behavior. Several examples of solving this problem are provided, most of which add complexity and a high degree of machine dependence (a counter goal of Ada). Unfortunately, forced solutions of this variety often predominate in complex computer-based systems development.

**Individual and group psychology.** That computer experts view problems based on their experience is quite natural. However, precisely this aspect constrains the development of appropriate, commonly agreed on philosophies. People react based on anchoring and adjustment; that is, by establishing an anchor point of view (for example, one or more of the computer-based artifacts) and then by making small adjustments from this anchor point. At the outset, the problem is viewed in terms of artifact constraints — negative thinking — instead of in terms of suitable problem-relevant solutions — creative positive thinking.

Anchoring and adjustment also exist at the group level, leading to the "not invented here" syndrome that prohibits positive development. Other group-related constraints, based on human interaction, include ego, personality clashes, previous education and training, prejudices, customs, and tradition.

When a common philosophy comes into existence, the group generally feels a lift and starts to attack problems with new enthusiasm. At the point the philosophy develops, they may not even know what is happening. The evolution of the philosophy may come during implementation, when group members gain insight through the programming process, as cited by Peter Naur.[7]

"All the world loves a winner" is another important psychological aspect of the situation. Often we are so fixed on succeeding that we do not learn from our failures. The fear of failure also drives projects into conservative, uncreative thinking. Wurman[8] regards the fear of failure as a general societal problem in which success is rewarded and failure is punished. Great designers in all fields have experienced successes and failures; most importantly, they have also learned to understand the positive aspects of failure.

**Management, formalism, and creativity.** After reviewing the international literature in software engineering, Lennartsson[5] reports three major areas of emphasis in complex software projects.

First is the emphasis on project management as the key to success. At the outset, the problem and potential artifacts are viewed as complicated, so a project management staff is established to deal with the complexity. In this environment, new artifacts (for example, CASE tools) are required to manage the selected artifacts, thus compounding complexity. This is called the management of complexity approach.

**Employers get what they ask for: artifact-bound employees who may or may not be able to solve the real problems.**

Others emphasize formalisms: that is, using linguistic notations and their semantics to, in a "Descartian spirit," specify and prove the correctness of a system. This altruistic view of seeing problems through a formal artifact has not resulted in widely accepted practice. On the other hand, formalisms that are well understood by the group and provide a good means of communicating a philosophy are extremely important. The reality related to notations and languages has been elegantly noted in *Compiler Construction: An Advanced Course* (Springer Verlag, 1976):

> The universe and its reflection in the ideas of man have wonderfully complex structures. Our ability to comprehend this complexity and perceive an underlying simplicity is intimately bound with our ability to symbolize and communicate our experience. The scientist has been free to extend and invent new languages whenever old forms became unwieldy or inadequate to express his ideas. His readers, however, have faced the double task of learning his new language and the structures he described. There has, therefore, arisen a natural control: a work of elaborate linguistic inventiveness and meager results will not be widely read.
>
> — William McKeeman

The final area of emphasis is creativity. Project success depends on supporting the creative, visionary people who can establish, communicate, and perpetuate the philosophy. The management must trust these creative people by providing them with authority as well as responsibility. This emphasis is the management of creativity approach.

The key to success is not based on only one of these approaches. In fact, each approach is relevant. However, it is important to begin with the creative aspects (that is, philosophy building) and then move into the other areas. In fact, a sound philosophy naturally provides appropriate views of formalisms and management structures.

**Cultural differences.** The societies in which computer-based systems are developed and used also play an important role in determining whether the development of a philosophy is considered important or is even considered at all. This can be affected by national traditions, the economic system, political views, education and training, views of marketing and supporting products, legal aspects, and other factors.

For example, Halang[9] indicated that Europeans use more higher-level artifacts — particularly higher-level languages — when implementing real-time systems. In contrast, Americans still have a highly pragmatic attitude, despite the fact that the largest, most complex real-time applications are designed and sometimes realized there, particularly in space and military systems. Americans often solve problems as quickly as possible with the artifacts available, even though documentation, system consistency, and other areas suffer. Halang points to traditions in European industry as a reason for this supremacy. Having lived, been educated, and worked for over 30 years in the USA and almost 20 years in Europe, I can confirm this difference of attitude, despite some notable counter examples on both sides of the Atlantic. This article's three examples of philosophies for real-time projects are all taken from European experiences.

We can observe cultural differences in project management, the use of formalism and creativity approaches, as described above. The management emphasis is predominant in North America, while the formalism emphasis is found in a few European projects. The creativity emphasis can be found in isolated cases all around the world; however, in the Swedish cases studied by Lennartsson,[5] the management of creativity was the dominant area of project emphasis.

**Organizational aspects.** The organizational environment plays a significant role in the development and/or use of computer-based systems. In the real-time area, the system may be developed to solve the specific problems of business operation, for example, power regulation and distribution by a power supplier or automatic train control by a national railway. The organization installing such systems invests in design, development, operation, and maintenance. The project may be totally in-house or subcontracted to other suppliers. However, it is vital that philosophy development either stays in-house or transfers with the product into the qualified end user's organization.

Let's now consider corporations that supply computer-based systems to both specific and general marketplaces. The product philosophy they develop must be viewed by and guide all parties with a vested interest in the product, namely, the development team, the management, the marketing organization, and the end users. The philosophy becomes the keystone of communication about the product.

It is also important to observe how computer-based projects are staffed, specifically, the qualifications of the people involved. A review of the classified advertisement sections of daily newspapers and professional publications indicates the predominance of artifact-related personnel advertisements. People are sought for their knowledge and skills in relationship to specific artifacts. While it does occasionally happen, there are too few advertisements specifically seeking people who can solve a problem or class of problems. Employers get what they ask for: artifact-bound employees who may or may not be able to solve the real problems.

A vital ingredient to successful projects is a stable work force. Here philosophy and esprit de corps can play a dominant role. The best methods and tools of software engineering will not contribute much if the work force is unstable. In this fluid environment of project personnel, the developments often degrade to highly pragmatic ad hoc solutions with all the long-term consequences that are implied. A sound explicit philosophy will encourage project members to stay with the project and insure that others can take over without major project perturbations when departures occur. Cultural differences as well as business operations and practices naturally influence work force stability. In the US, which has a highly volatile market for qualified computer professionals and engineers, this aspect has influenced many important projects, undoubtedly contributing to incomplete and/or unsatisfactory problem solutions. In Europe and Japan, project groups tend to be more stable.

In the following sections, we'll consider three concrete examples of how philosophy development has played a key role in successfully implementing real-time systems. The goal is not to be complete in any sense; however, I provide the problems to be solved, the main concepts and ideas related to each project philosophy, and the impact of the philosophy on further project developments and related parties.

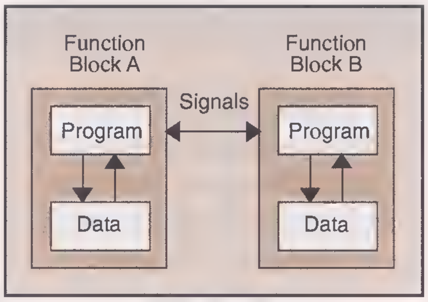


**Figure 1. Blocks and signals.**

## The Ericsson telecommunications experience

The public telecommunications division of the LM Ericsson corporation, based in Stockholm, has a long history as a supplier of telephone and telecommunication equipment to both the Swedish PTT (Televerket) and the rest of the world. Today, Ericsson telephone switching equipment is installed in more than 70 countries worldwide. It is the world market that provides Ericsson with an economic basis for the advanced research and product development required for leadership in the telecommunication industry.

In the late 1960s, the company decided to move into the era of digital stored-program-control (SPC) switching equipment. Since its market was diverse, with much local adaptation required for its products and with stable long-term solutions needed (20- to 40-year product life time), Ericsson sought to develop an appropriate philosophy for its SPC exchange products.

The designers had an appreciation of computer technology but no long-term experience with it. (They were very familiar with conventional relay techniques.) This lack of deep attachment to computer artifacts probably was an important advantage. Instead of viewing the problem through the limitations of specific commercially available computer artifacts, they attacked the problem from the points of view of both their experience and the market requirements.

The design effort led to the concepts of blocks and signals as abstract objects that represented the fundamental components of the switching equipment. The concepts were a direct extension of the designers' experience with the previous analog systems. The two concepts originated during the development of the first Ericsson SPC product called AKE. These concepts were later formalized — that is, given proper semantics — and propagated to the very successful AXE switching system philosophy. Figure 1 illustrates these concepts.

There is a strict difference between programs and data. The data of a block is only accessible from within the block, thus providing security and protection. The signals range from simple signals to messages that are transferred between function blocks.

Unfortunately, very little has been published concerning the Ericsson philosophy of stored program control exchanges. However, Jacobson[10] reports in his thesis the following rationale for the block and signal philosophy:

> (1) Blocks are manageable units for the design, production, installation, operation, maintenance, etc. of large systems. This is a software engineering aspect of blocks, making possible the division of work on a large system into parts that can be planned, worked, tested, produced,... separately and then integrated as a system.
>
> (2) Blocks are units of encapsulation. The only means of accessing the internals of a block is through a strictly standardized signal protocol. From outside the block only the signal protocol is visible, the internal structure and implementation being hidden to the user of the block.
>
> (3) Signals offer dynamic interconnection of blocks. For instance a given block A can be interconnected to many different other blocks, but in a given situation the receiver block B is dynamically known to A as a data object. When A sends a signal to B, the actions taken is decided solely by the receiver block B, and the sender block A specifies not more than the signal name and the data object referring to B.
>
> (4) Blocks can be implemented using different techniques for different blocks, such as different programming languages, different computers or computer systems, different hardware techniques, and so forth. Block decomposition therefore supports adaptation to new technology without redesign (or with limited redesign).

Blocks and signals are the components that the designers of particular installations work with. Since many functions are common, libraries of blocks are developed and reused or modified as needed. New technical telecommunication employees are indoctrinated into this natural Ericsson SPC philosophy and way of doing business.

As stated in point (4) above, an important aspect of the project was the separation of logical function from physical implementation, that is, philosophy from carrier. The original designers had the foresight to keep these notions clear. Thus, blocks and signals can be implemented via hardware, software, or combined components. Further, as new hardware becomes available, it can be incorporated into AXE systems by implementing the block and signal philosophy. In the first AKE realization, block and signal concepts were implemented via assembly language macros. The AKE project provided important positive and negative learning experiences. For the first version of AXE, a special-purpose CPU was developed to conveniently map the more formal version of the block and signal concepts to a programmed control computer. Further, the block and signal concepts were embedded into a special-purpose programming language for the processor called PLEX. The general structure of this architecture is illustrated in Figure 2.

The architecture is composed of a microprogrammed CPU and three separate storages. Each block has a unique block number, reflected in the block number register. This register points to the reference store where the start address of the block program code in the program store and a base address table for referencing the data store are found. The instruction register is used to sequence instruction execution within the block program code in a conventional manner. Further, a pointer register is used to reach individual data in a variable. Thus, the architecture as well as the tools (that is, artifacts) used to realize the SPC philosophy were a natural follow-on development from the basic philosophy.

Each AXE exchange contains one central and several regional processors. Blocks are the basic program units for both processor categories. One example of the consequence of a consistent philosophy is the question of call synchronization. Blocks operating in the regional processors cannot signal each other directly; they must use the central processor, which manages the synchronization. The block and signal philosophy permitted this architectural property to be conveniently implemented and controlled by software tools. In fact, a host of design support tools were developed, including PLEX, with related software engineering support for the life-cycle management of the SPC products.

Development of AXE transpired at Ellemtel, a research and development corporation jointly owned and operated by Ericsson and the Swedish PTT (Televerket). There were a few key people who created and perpetuated the block- and signal-based philosophy; namely, Bengt-Gunnar Magnusson (AXE general hardware architecture), Ivar Jacobson (AKE software architecture), and Göran Hemdal (AXE software architecture). As with many pioneering efforts, the proposed project philosophy was first viewed with skepticism. Eventually, full support was given and the AXE product was successfully developed at Ellemtel. The management of creativity approach proved to be a key to success.

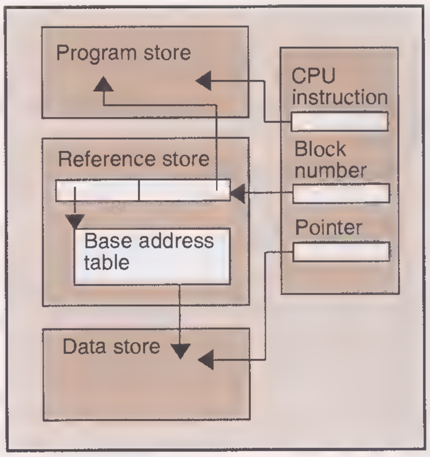


**Figure 2. CPU architecture for the AXE system.**

The philosophy of AXE has affected all aspects of the Ericsson telecommunication division activities. From the marketing point of view, it has been the basis for convincing customers that Ericsson indeed has a long-range approach to developing and supporting switching equipment. The separation of logical design from physical implementation permits the system to be modernized during the long life cycle. This represents a major advance over the older, more conventional relay technology. One highly successful adaptation was the incorporation of cellular mobile radio communications into AXE exchanges. This adaptation opened a new, rapidly expanding market for both Ericsson technologies.

The AXE philosophy has led to an esprit de corps in the company. More than 3,000 Ericsson employees are actively involved in developing and supporting AXE switching products worldwide. Finally, AXE has provided the Swedish PTT with a state-of-the-art product that enables Sweden to be a world leader in advanced telecommunication equipment. Sweden, in particular, and the Nordic countries, in general, have the highest ratio of telephones, computer terminals, and mobile telephones per capita in the world.

**A counter example.** On the other side of the Atlantic, during the mid-1960s, Bell Telephone Laboratories at Indian Hill, Illinois, actively pursued the same goal of developing stored program control switching systems. In contrast to the Ericsson approach, the designers started by accepting the use of an IBM 7090 series computer (that is, an artifact). Further, as the design developed, it was implemented in the computers' assembly language (that is, another artifact). The implementors' primary view of the switching system became the assembly language code. The result, as one would expect, was a prime example of spaghetti code, and the approach was eventually abandoned. It did, on the other hand, serve as a costly but useful learning experience for Bell Telephone Laboratories.

I have been a consultant in several computer-based projects for both qualified end users and suppliers in which my major role has been to assist in developing appropriate philosophies. In the following sections, we'll consider two of these projects. Generally speaking, consultants are called when problems (frequently severe) have been encountered. The consultant is then viewed as the fireman engaged to put out the fire. The philosophy-lacking project is often well established, making it difficult to create quick fixes. It is better to back out, understand the mistakes, take a good long look at the problem, retrench, and build up the philosophy. The first project (Automatic Train Control) was of this nature. The second project (High Voltage Power Dispatching) was accomplished through a philosophy-first approach. There I participated in the project from the start. A very competent colleague, Miguel Bertran, and I developed a philosophy that guided the successive stages of system development.

## Automatic Train Control

The Swedish National Railways in the mid-1970s decided to implement a control system to help train engineers follow speed

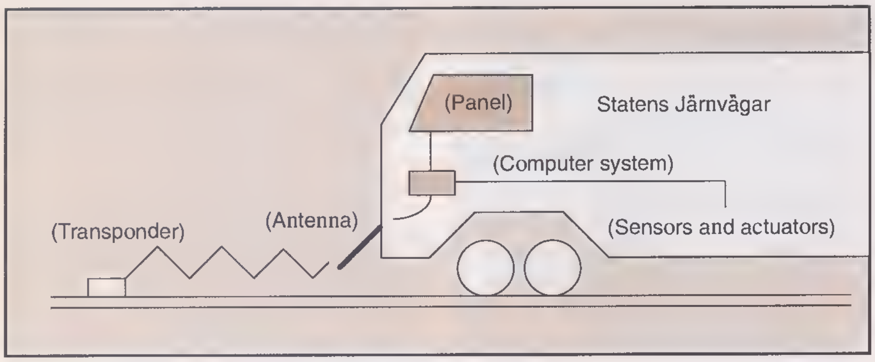


**Figure 3. Major "components" of the ATC system.**

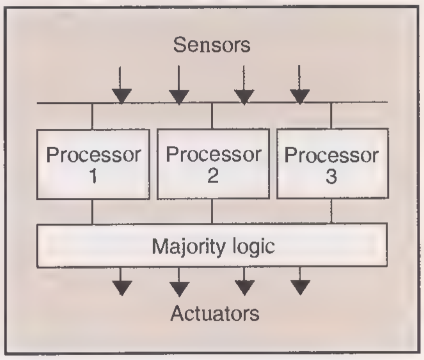


**Figure 4. View of the ATC on-board computer system.**

limits along the railway system. The Automatic Train Control (ATC) system contains two parts, one for telecommunication and one for the operator. The telecommunication part involves radio transmission from transponders located periodically on the tracks to the locomotive. Read-only information concerning speed limits, position, distance to next transponder, and so forth is transmitted to the train as it approaches the transponder. The computer control and operator part of ATC, located in the locomotives, monitors and controls the train and communicates with the train engineer(s). It is a "watchdog" system that advises the engineer and only stops the train or reduces its speed when the engineer fails to do so. Figure 3 shows the major components of the ATC system.

The contract for the telecommunication part was awarded to the Ericsson corporation, and the computer control and operator part was awarded to Standard Radio and Telefon (at that point owned by the ITT Corporation). Both corporations operate in Stockholm. I consulted for the computer control part. An implementation philosophy led to an operating solution that clearly placed the development "on track."

A highly competent engineer at Standard Radio had been in charge of the project about one year when I arrived. The engineer had developed a significant amount of assembly code for a PDP 15 computer that simulated the ATC system. He started writing code at an early stage, even before the functionality of the system was clear. While he was a very clever programmer, the solution was unmanageable and was rapidly moving toward spaghetti code. Fortunately, he recognized this fact himself and was quite cooperative in working with me to retrench and develop an appropriate philosophy.

The on-board system is composed of triplicated microprocessors surrounded by sensors and actuators, as well as operator communication. All three systems perform identical processing. Calculated results are checked by majority logic before they are delivered as system outputs. A simplified view of the computer system is shown in Figure 4.

After studying the problem, the environment in which development and testing would take place, and the problems for future updating and maintenance, I suggested a philosophy for implementing the real-time operating system that corresponded quite closely to the nature of the problem, including the use of limited-capacity microprocessors. The philosophy was based on the requirement for a continuous system with relatively few required processes. This led to the idea of viewing the system as continuous, cyclic, precisely-timed loops.

The original proposal divided a major execution cycle of 250 milliseconds into five time frames: A,B,C,D, and E. During A, C, and E, decoding transponder information transpired; whereas, B and D were dedicated to the primary processing, including speed and distance calculation, brake pressure updating, speed limit button handler, speed limit scheduler, stop check, presignal braking, prebraking and deceleration supervision, speed supervision, and output calculations. This original proposal was further simplified by treating all processes as members of a single 250-millisecond cycle. The predicates that determine if the process is relevant during the current cycle are tested by a guard at the head of the process. A full cycle will always complete in a maximum of 250 milliseconds. If it finishes ahead, it will wait for a clocked signal to start the next cycle.

With this simple structure, many aspects of the project became quite clear. Some central aspects that evolved were

- A simple loop of procedure calls provided the backbone of control.
- The program assembly code was structured around the simplified process notion and control structure.
- Sufficient basic loop performance reduced the need for immediately handling hard interrupts.
- Interprocess communication was accomplished by shared process variables where the producer was executed before the consumer(s) in the loop.
- The system was deadlock free.

The proposed philosophy was gratefully accepted, and development was restarted with very positive results. I proposed plans based on the philosophy for production control, testing, verification, updating, and maintenance of the system. Several tools were suggested that would produce code in a readable manner and permit consistent, automatic checks by parsing the assembly language source text. The philosophy led to intuitive and consistent solutions to problems that arose during project implementation.

The net result was that the Swedish National Railways received a highly reliable, efficient, and maintainable system. The system has been installed in over 1,400 locomotives in Sweden. Variations of the system have been exported to other countries, and, according to Standard Radio and Telefon management, the system has become a showcase for implementing automatic real-time control systems for trains.

Many are extremely surprised that the system only requires 8K of memory. At that time, and even today, the low memory volume contributed to lower hardware production costs and a small physical structure. A new generation of the product is currently under development where the memory requirements will be extended to 16K, very modest in these days of megabytes of software complexity.

The same ATC philosophy is being used for planning control of high-speed traffic trains (up to 300 kilometers per hour) that will be installed during the 1990s in Sweden. The philosophy is viable and lives on.

## ENHER: Power Dispatching

The final project involved the qualified end-user development of the central system for a high-voltage power dispatching center at the ENHER (E.N. Hydro Electrica del Ribagorzana) power company in Barcelona. The name given to the project was Conce (Control Central). In contrast to the short period of time I was involved in helping Standard Radio (one year), the work with ENHER proceeded over six years (1974-1980). I had the pleasure of working with Miguel Bertran, a former graduate student and then project leader at ENHER. Together, we developed a philosophy for the implementation of the central control that naturally led to developing and using a variety of appropriate software engineering methods and tools.

After examining the existing commercially supplied systems for high-voltage power dispatching, the management at ENHER decided to build their own system. Their motivation was cost; the purchase, installation, training, operation, and maintenance requirements for any commercially supplied systems would involve substantial investments. At the same time, only a limited competence in respect to any purchased system would exist in-house. The management believed that it would be better off with a system for which detailed competence existed in its own organization (that is, by being a qualified end user). This situation provided a strong motivation for developing a philosophy that would touch many related parties.

When we started the Conce project, several papers appeared concerning monitors, which are quite important in solving problems such as readers/writers, controlled access, resource allocation, communication, and monitoring. These papers influenced our thinking in developing the central philosophy. Even though we knew that, for practical reasons, the system would be implemented in Fortran with a manufacturer-supplied operating system (artifacts) and hardware, we concentrated on developing a philosophy that matched the real problem. Thus, we built the philosophy around processes and monitors, as shown in Figure 5. The monitors provided orderly system access to important resources, such as the network, the operators, the power grid information stored in central matrices, and the structured use of application processes.

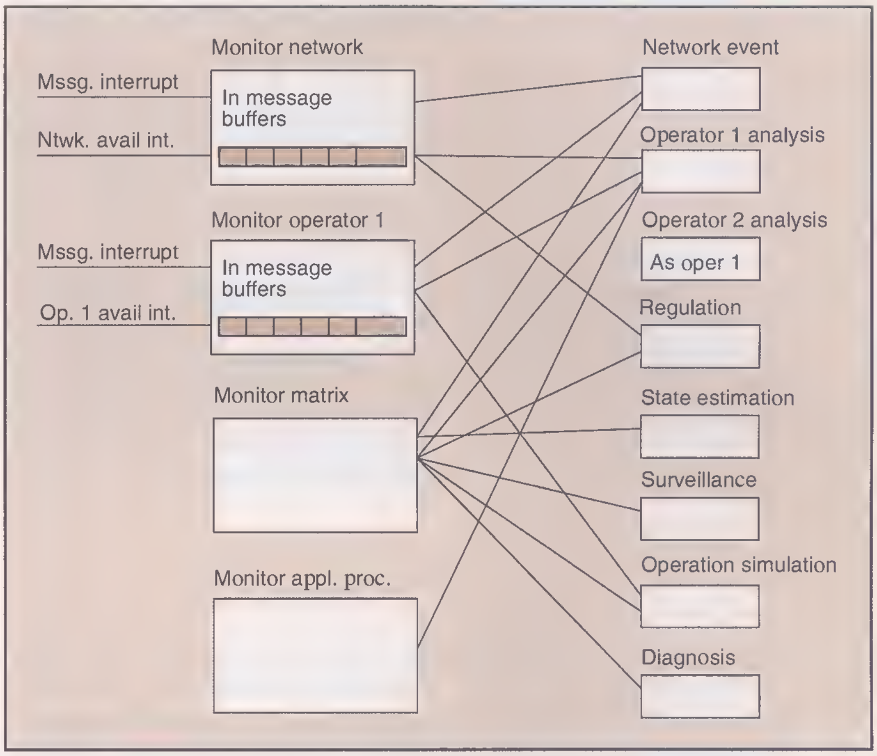


**Figure 5. Monitors and application processes of the Power Dispatching System.**

The monitor and process philosophy led to some extremely important properties for the further development of the system:

- We could start developing monitors and subsets of the application processes and test the framework in partial form.
- The monitors provided a useful clearinghouse for instrumenting the system testing, statistics gathering, and performance analysis.
- The philosophy led to simulating partial versions of the system, resulting in gained insight and experience.
- We could successively add details and gain further insight by stepwise refinement.
- System development became a process of implementing successively complex simulators (models) where the final simulated version became the real system.
- The system's logical structure could be used in requesting equipment and system software with related price information from computer manufacturer suppliers (request for price quotation, or RPQ).

After we gained insight and experience with the project, we developed a configuration specification reflecting project needs and supporting the philosophy. The logical configuration is shown in Figure 6.

The logical system structure includes three CPUs; the primary role of each of the three computers (operation, hot standby, and development) is shown in the figure. However, since the memory is shared and all other system connections are controlled by program-changeable switches, the three roles are interchangeable. One computer always runs the system, at least one (possi-

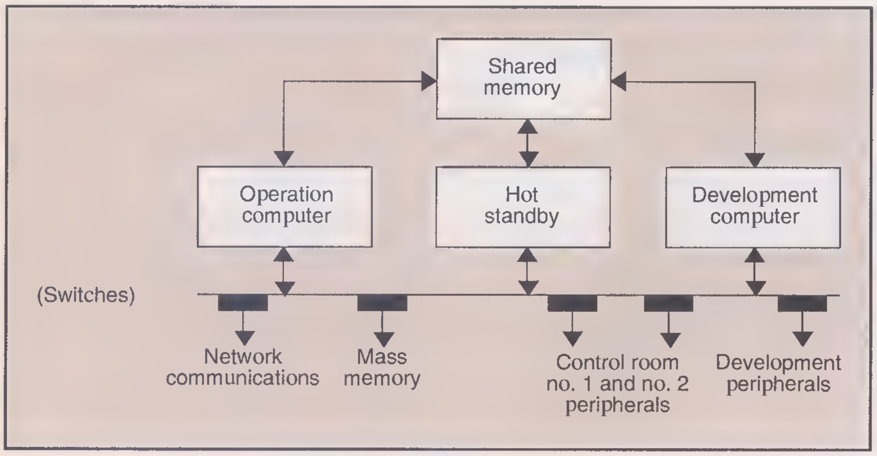


**Figure 6. Logical configuration of the Power Dispatching System.**

bly two) is in hot standby, and one (when needed) is used for development. Because the development system also has read-only access to the real power network information, simulated new versions of the system could be made before switching them into operation.

This logical system specification was supplied to six potential suppliers with a request that they respond to the concrete system details for the logical configuration. This RPQ approach surprised most of the sales people because it was not standard practice. While responses left quite a bit to be desired, the information received could be evaluated in a structured manner. After evaluating the technical and practical matters, we selected an Interdata system.

A degree of orthogonality naturally existed between our philosophy and the manufacturer-supplied hardware and system software; however, we resolved these differences. The monitor approach provided a natural means of extending the philosophy to the shared resources of the computer system. In addition to the monitor structure we had worked with earlier (Figure 5), we developed monitors for the remaining resources in the physical configuration.

As a result of the philosophy, we selected some existing congenial methods and tools and developed others that were appropriate for the philosophy. To provide clear, communicable documentation of procedural programs, we used the dimensional flowcharting (DF) technique developed by Witty.[11] This technique was a valuable method for documenting and communicating program logic.

A further development lifted the abstraction level of describing certain application processes, in particular, the operator communication and the power grid database. We developed syntax-directed methods for these activities. We constructed a tool for recursive descent parsing and for generating Fortran programs from TBNF (a modified BNF grammar) syntactical descriptions. The tool, called translator basic (T-Basic), provided formalisms for both semantic and syntactic descriptions. From this tool, many nontrivial application processes were automatically generated into Fortran source code. T-Basic was also used to produce source program filters that generate corresponding DF representations. Further, the DF graphical form also was an excellent way to structure and communicate the grammar to group members.

The project team grew successively from 1975 to 1980. When each new project member joined the group, the philosophy was made clear both abstractly and concretely. The DFs and syntax descriptions provided a straightforward manner to acquaint new members with the philosophy and important system functions. A PC-based real-time simulator driven by DF descriptions was developed and was used to estimate the timing impact of additions to the existing system.

The Conce project development proceeded in this general order: philosophy building, logical structure, methods and tools, preliminary simulation, equipment specification and purchase, implementation by successively complex simulators, and, finally, installation. This succession is an appropriate model for a wide class of qualified end user-developed computer-based systems. A few papers were published during the project that reflect the project philosophy, for example, Bertran and Xampeny's summary paper.[12]

## Improving communication and understanding

The major thrust of this article has been to emphasize the importance of developing project-relevant philosophies to guide successful computer-based projects. The bottom line is that the philosophy makes major contributions to human-to-human communication about the computer-based system. It improves the understanding of a wide range of parties with vested interests in the project. Thus, a philosophy improves all aspects of the products' life cycle and, as shown in this article, leads to successfully implemented projects.

**Common denominators: blocks, classes, processes, and objects.** If we examine the three projects and the Simula experience, a set of related concepts appears: structuring around blocks, classes, processes, and objects. In practice, a detailed semantic definition of these concepts, while useful, is not as essential as a common understanding of the principles. Ivar Jacobson, one of the early contributors to the Ericsson Telecommunications AXE philosophy, has further refined the philosophy and is now actively involved in spreading a modern version under the name Objectory. This represents a major break from the CASE approach of marketing artifacts. Jacobson quite correctly believes that the philosophy is the most important aspect and that the artifacts, including programming language and design support tools, are important but secondary consequences of the philosophy.

Object-oriented design (OOD) is now in vogue, although the ideas and their implementation have existed for quite some time. The use of object as an abstract unit moves us closer to the real objects of the problem to be solved. OOD concepts, in one form or another, will greatly influence computer-based system development in the 1990s.

**Representation versus philosophy.** Compared to a sound philosophy, how much do linguistic or graphic representations of a problem contribute to communication and understanding? This question is, of course, virtually impossible to answer quantitatively. I assert that philosophy is the major contributor and that linguistic and graphic representations play important but secondary, supporting roles.

Many are lured into thinking that common languages for specification, design, or implementation are the key to improving communication and understanding. This is particularly prevalent among those who practice the management of complexity approach. Naturally, common languages contribute to communication possibilities, but it is still the philosophy that is of primary importance. The representation is an important but secondary matter. In Ada, for example, the philosophy aspects of packages, generic procedures, and rendezvous provide more understanding than the concrete aspects of representing programs.

**Requirement analysis, specifications, and prototyping.** In all projects, the resultant product must match users' needs and wishes. Naturally, clear requirements statements and specifications are vital to a common understanding between users and implementors. However, an overemphasis on these two issues can be symptomatic of the management of complexity approach. A sound philosophy supports and expedites requirements analysis and specification. If a problem-relevant philosophy does not exist for the project from the start, it should be developed as soon as possible since it guides all further activities. After establishment, the philosophy must be nurtured and propagated.

Exploring plausible approaches to developing computer-based system applications can be expedited through prototyping. The area of rapid prototyping is receiving considerable attention. While it can provide insight, it only provides models of some required behaviors and technical aspects of the project. Thus, a prototype is not necessarily a complete or sufficient basis for assuring successful implementation. Further, much of the current work on prototyping is based on developing special-purpose languages for prototype construction and analysis. The prototyping language requires further knowledge and understanding, with all of the project costs and complexities that are implied. The final product is typically developed using a different programming language representation, even though there are some prototyping languages that permit automatic or semiautomatic translation to common implementation languages.

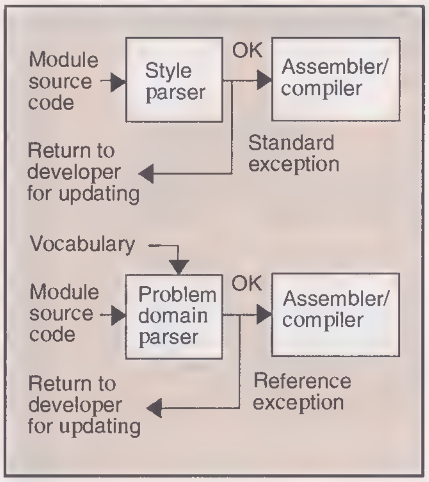


**Figure 7. Source program filter examples.**

A proven alternative to rapid prototyping is staged partial development, also referred to as incremental development. By this I mean an initial superstructure designed and constructed in the implementation language. The first version provides critical central functions. In the first version, however, application functions are excluded. Most importantly, the superstructure is "instrumented" with information gathering and analysis facilities. The superstructure is simulated and evaluated. In stages, new service functions and application functions are added and simulated. Within the framework, new requirements are added along the way. The project converges towards a suitable solution with continual feedback and adaptation after each stage. Continual, simulated user interface evaluation is an essential part of the staged development process. When all desired functions are implemented, a version of the system that bypasses the instrumentation facilities is generated. The final simulated version is the real version. The instrumented version, however, is vital for continued development, evaluation, and maintenance.

The development process I have described is exactly the sequence of events used in developing the Fortran-based real-time system for power control at ENHER. The philosophy based on monitors and application processes provided an excellent starting point for successively developing and evaluating simulated versions leading to the installation and operation.

**Unified set of problem-relevant methods, mechanisms, and tools.** The three examples cited in this article illustrate how software engineering methods, mechanisms, and tools evolve as a consequence of the philosophy, not as the driving factor determining system development. This evolution leads to sets of unified, highly congruent methods, mechanisms, and tools that provide many practical and economic advantages, especially in communication and understanding. The report[5] on other Swedish projects confirms the success of this approach. Further, Lennartsson points out that locally developed supporting artifacts have not involved major development costs.

To illustrate a few inexpensive tools, let's consider verifying that source programs abide by both style and problem domain-relevant design rules. Parser generators are one of the best known and developed program automation tools availahle. It is a straightforward matter to produce filters that verify various aspects of source texts before their assembly or compilation. Consider the two parsers illustrated in Figure 7.

The style filter assures that general project programming conventions are followed. These conventions come from within the project group and reflect both their current needs and the needs of future viewers of the source programs. For example, the style filter can include examining the structure and contents of comments, examining the naming of variables, procedure names, indentation conventions, and so forth in the module of code to be assembled or compiled.

The problem domain filter, for example, can verify that the program module uses variables in a consistent manner (for example, read, write, read/write, execute capabilities). The identification of problem domain objects is entered into the parser as a vocabulary. These simple, inexpensive tools catch a large number of potential mistakes and achieve a reasonably high degree of verification.

The suite of methods and tools typically includes specific project tools (designed and normally implemented as a part of the project), as well as tools supplied by manufacturers or software vendors. External tool selection must be based on the degree of consistency with the project philosophy and on the education and training required for their use and continued support.

**Reduction of documentation requirements.** A sound philosophy which has driven large and small project decisions and continues as an integral part of the philosophy reduces documentation requirements. This avoids voluminous details placed in folders or retained in large files that are rarely read and costly to maintain.

**Moving the artifacts closer to the problem.** I have exemplified the use of philosophies in computer technology and the practice of computer-based application realization. The philosophies contribute to a general improvement; however, there still remains a dichotomy between the nature of the problems and the artifacts. Even though blocks, classes, processes, and objects provide useful abstractions of the real system objects, they are most often concretely realized by procedural code. Thus, in the end, we are required to program as well as to communicate the details of the solution in a procedural programming language. Object orientation has moved us closer to the nature of the problem; however, it is important that we continue to explore ways to come even closer.

For example, in the real-time domain, we could consider using differential calculus to characterize the behavior of continuous time segments and using logic to characterize discontinuities. This would be a natural way of viewing real-time phenomena. Eventually I hope we will move into an era where concrete procedural programming as we know it today can largely be avoided, even for complex real-time applications. Then, the fundamental physical problems (typically analog) can be expressed in terms of relevant mathematical and physical properties. These are the goals of a Swedish effort investigating a new approach for building upon the principles of functional- and logic-based languages, temporal logic, and a holistic view of hardware and software architectures. The goal is to move the entire set of artifacts used in real-time system projects closer to the problems.

**Design for understandability.** Understandability is introduced here as a property of design in the same sense "testability" is used in relationship to "design for testability." As mentioned several times, a clear problem-relevant philosophy leads to improved understanding. However, everyone who adds value to a computer-based product must design for understandability. Design for testability is an important area for integrated circuit design. To be testable, a design must be understandable. However, as we have considered in this article, understandability is a much deeper issue for all life-cycle aspects of computer-based products.

Understandability is relevant only to the receiver of information. People interpret and understand new information in terms of what they already understand.[8] Thus, one single, universal means of communicating the structure and implementation of computer-based systems will always be an illusion. Various informal and complementary formal descriptions are required to instill understanding for various receivers. This dual approach can be quite useful. The formal description, however, must be understandable by the receiver; otherwise, the natural control noted by McKeeman, cited earlier in this article, will take effect:

> There has therefore arisen, a natural control: a work of elaborate linguistic inventiveness and meager results will not be widely read.

The understandability of computer-related products, in general, and computer-based products, in particular, will become an issue of increasing importance in the future. This is true for reasons of safety, accountability, responsibility, and legal liability for products. Since complex computer-based systems continue to explicitly and implicitly penetrate our daily activities in increasing variety, I believe that the world will not continue to accept the prevailing buyer beware attitude shown by the hardware and software producers and suppliers. ■

## Acknowledgments

I would like to acknowledge and thank all colleagues who over the years I have had the pleasure of working with to develop philosophies for computer-based systems. With regard to the personal real-time system experiences cited in this article, in particular, I thank Sivert Wallin, Berit Bryntse, and Roger Andersson for their cooperation and support in developing the philosophy used in the Automatic Train Control system. Sivert Wallin reviewed this article and provided an up-to-date status of the project. On the ENHER high voltage power dispatching control system, I express appreciation to Miguel Bertran, who made the philosophy become a reality at ENHER. I also thank the project manager, J. Xampeny, who patiently listened, understood, and supported the development of the philosophy.

For reviewing the paper and presentation of the AXE philosophy, I'd like to thank Ivar Jacobson, formerly of Ericsson and now at Objective Systems AB, Stockholm, and Bengt-Gunnar Magnusson, who established the hardware architectural model and led the project development at Ellemtel.

Gunnar Carlstedt and Erik Tengvald of Carlstedt Elektronik AB, Göteborg, provided constructive comments on this article and have been instrumental in showing the way toward a futuristic real-time system philosophy. Tengvald of Carlstedt Elektronik AB and Erik Sandewall of Linkiping University provided insight into futuristic philosophies for real-time systems. Bengt Lennartsson of Linköping University reviewed the work and contrasted it with his own study of similar projects in Swedish industrial environments. Christer Scheja and Peter Hansson at Tour och Andersson Industri Automation AB, Malmö, have stimulated my interest in producing this article. I thank Ingemar Carlsson and Ingvar Åkersten of the Swedish Defense Material Administration (FMV), as well as Ingemar Ögren, consultant, for their encouragement and reviews of this material.

Finally, I thank the referees for their constructive criticism of the contents of this article and for their encouragement to present the arguments concerning philosophy to the wider range of computer-based systems with real-time systems used a subset for concrete illustration.

This work has been partially supported by Tour och Andersson Industri Automation AB (TAIAB) in Malmö and by the Swedish Defense Material Administration (FMV), Stockholm.

## References

1. F.P. Brooks, "No Silver Bullet: Essence and Accidents of Software Engineering," *Computer*, Vol. 20, No. 4, April 1987, pp. 10-18.

2. O.J. Dahl, B. Myhrhang, and K. Nygaard, *Simula 67 Common Base Language*, Norsk Regnesentral, Oslo, Norway, 1968.

3. H. W. Lawson, "PL/I List Processing," *Comm. ACM*, Vol. 10, No. 6, June 1967, pp. 358-367.

4. G. Radin, "The Early History and Characteristics of PL/I," *Proc. History of Programming Languages Conf.* (ed. Wexelblatt), *SIGPlan Notices*, Vol. 13, No. 13, August 1978.

5. B. Lennartsson, *Systemkonstruktion i programvara - några svenska exempel (Software Systems Design - Experiences from Projects in Swedish Industry)*, Mekanförbundet, November 1988 (in Swedish).

6. T.P. Baker, and A. Shaw, "The Cyclic Executive Model and Ada," *The Journal of Real-Time Systems*, Vol. 1, No. 1, June 1989, pp. 7-25.

7. P. Naur, "Programming as Theory Building," Microprocessing and Microprogramming, *The Euromicro Journal*, Vol. 15, No. 5, May 1985.

8. R.S. Wurman, *Information Anxiety*, Doubleday, New York, 1989.

9. W. A. Halang, "Education of Real-Time Systems Engineers," Microprocessing and Microprogramming, *The Euromicro Journal*, Vol. 25, No. 1-5, January 1989, pp. 71-75.

10. I. Jacobson, Concepts for Modelling Large Real-Time Systems, Doctoral Dissertation, Department of Computer Systems, The Royal Institute of Technology, August 1985.

11. R. Witty, "Dimensional Flowcharting," *Software Practice and Experience*, Vol. 7, 1977, pp. 553-584.

12. M. Bertran, and J.A. Xampeny, "Computerized Power Network Telecontrol Center: Environment and Solution Framework," *Proc. IEEE Power Engineering Society Summer Meeting*, Vancouver, Canada, July 1979.

**Harold W. Lawson** is an independent consultant. He has been active in the field of computing since 1958 and has broad international experience in industrial and academic environments.

During his industrial career, he contributed to several pioneering efforts in hardware and software technologies at Univac, IBM, and Standard Computer Corporation. He has held permanent and visiting professorial appointments at several universities and is an active consultant in North America, Europe, and the Far East. His publications include several books and contributed chapters, as well as over 60 technical contributions. He has been active in many professional societies and is a senior member of IEEE.

Lawson received the BS degree from Temple University and the PhD degree from the Royal Technical University, Stockholm.

The author can be contacted at Lawson Publishing and Consulting, Inc., Torshammarsvägen 11 3tr, 181 33 Lidingö, Sweden.

**CALL FOR PAPERS**

*Individual Papers and Group Sessions are invited.*

**Northcon®/91**

***The Premier Electronics Industry Conference of the Northwest October 1-3, 1991 in Portland, Oregon***

*Northcon is bringing this comprehensive electronics conference and exhibition to Portland, Oregon, where more than 9,000 electronics engineering professionals - design, test and manufacturing engineers, specifiers, purchasing specialists, engineering management, R&D and corporate personnel, quality specialists and corporate executives - will gather to learn about the latest electronics products and technology.*

*Papers for presentation in the technical sessions are requested in five areas:*

1. *Design/Test/Production/Management*
2. *Computer Hardware/Software Advances*
3. *Research and Development*
4. *Quality and Reliability*
5. *Regulations and Environment*

*For a paper to be considered, a 1000–word summary must be submitted that states the objective of the paper, the new contributions, and the conclusion that will be made. Previously published materials are not acceptable.*

*Abstracts must be mailed or faxed no later than March 15, 1991 to be considered for evaluation.*

*Please submit abstracts to:*
*Jon S. Potts*
*Technical Programs Chair*
*c/o NORTHCON/91*
*8110 Airport Boulevard*
*Los Angeles, CA 90045-3194*
*(800) 877-2668 or*
*(213) 215-3976, ext. 251*
*FAX: (213) 641-5117*

Northcon®

is a joint venture of:

Portland and Seattle Sections of the Institute of Electrical and Electronics Engineers, IEEE

Electronics Manufacturers Association, EMA

Cascade Chapter of the Electronics Representatives Association, ERA

# FTCS 21

SYSTEMS • SOFTWARE • HARDWARE • PRACTICAL RESULTS • RESEARCH

## The Twenty-First Annual International Symposium on Fault-Tolerant Computing

**25-27 June, 1991**

**Le Grand Hôtel**
**777, rue Université**
**Montréal, Québec**
**Canada H3C 3Z7**
**Phone: 514-879-1370**

**Sponsored by:**
**IEEE Computer Society**
**In co-operation with:**
**IFIP Working Group 10.4**

The **Fault-Tolerant Computing Symposium** has become the major international forum in all aspects of fault-tolerant computing, such as: specification, design, implementation, test, diagnosis and evaluation of dependable and fault-tolerant computing systems. The scope of the symposium spans hardware, software and system issues.

Major topics include, but are not limited to: fault-tolerance in real-time systems, certification of safety-critical systems, software fault tolerance, fault recovery in transaction-processing systems, data security and integrity, fault tolerance in communication networks, validation of VLSI designs and implementations, testing and defect tolerance.

**About Montréal:**
Montréal, founded in 1642, is the only French metropolis in North America, and is the second-largest French city in the world after Paris. Located on an island in the narrows of the St Lawrence river, with a population of some 2 million, Montréal offers visitors a unique blend of American and European culture, with a "down home" flavour rooted in a rich folk tradition.

Symposium participants are invited to extend their visit for a few days to catch the Montréal International Jazz Festival, prowl the cobbled streets of Old Montréal, or just lounge in a bistro on rue St-Denis. Experience a city with a difference.

**Important Information:**

| Advance Registration | Canadian Dollars |
|---|---|
| May 15, 1991 | |
| Member IEEE | $375 |
| Non-Member | $475 |
| Student | $100 |
| At Conference Registration | |
| After May 15, 1991 | |
| Member IEEE | $450 |
| Non-Member | $550 |
| Student | $125 |

FTCS-21 Registration
c/o Prof. Janusz Rajski
McGill University
Dept. of Electrical Engineering
3480 University St., Room 633
Montreal, CANADA H3A 2A7
FAX: (514) 398-4470
email: rajski@spock.ee.mcgill.ca

IEEE COMPUTER SOCIETY ®

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

# An Overview of Common Benchmarks

**Reinhold P. Weicker**
**Siemens Nixdorf Information Systems**

The main reason for using computers is to perform tasks faster. This is why performance measurement is taken so seriously by computer customers. Even though performance measurement usually compares only one aspect of computers (speed), this aspect is often dominant. Normally, a mainframe customer can run typical applications on a new machine before buying it. With microprocessor-based systems, however, original equipment manufacturers must make decisions without detailed knowledge of the end user's code, so performance measurements with standard benchmarks become more important.

Performance is a broad area, and traditional benchmarks cover only part of it. This article is restricted to benchmarks measuring hardware speed, including compiler code generation; it does not cover the more general area of system benchmarks (for example, operating system performance). Still, manufacturers use traditional benchmarks in their advertising, and customers use them in making decisions, so it is important to know as much as possible about them. This article characterizes the most often used benchmarks in detail and warns users about a number of pitfalls.

**"Fair benchmarking" would be less of an oxymoron if those using benchmark results knew what tasks the benchmarks really perform and what they measure.**

## The ubiquitous MIPS numbers

For comparisons across different instruction-set architectures, the unit MIPS, in its literal meaning of millions of instructions per second (native MIPS), has lost nearly all its significance. This became obvious when reduced instruction-set computer architectures appeared.[1] Operations that can be performed by one CISC (complex instruction-set computer) instruction sometimes require several RISC instructions. Consider the example of a high-level language statement

$A = B + C$ /* Assume mem operands */

With a CISC architecture, this can be compiled into one instruction:

```
add    mem (B), mem (C), mem (A)
```

On a typical RISC, this requires four instructions:

```
load   mem (B), reg (B)
load   mem (C), reg (C)
add    reg (B), reg (C), reg (A)
store  reg (A), mem (A)
```

If both machines need the same time to execute (not unrealistic in some cases), should the RISC then be rated as a 4-MIPS machine if the CISC (for example, a VAX 11) operates at 1 MIPS? The MIPS number in its literal meaning is still interesting for computer architects (together with the CPI number — the average number of cycles necessary for an instruction), but it loses its significance for the end user.

Because of these problems, "MIPS" has often been redefined, implicitly or explicitly, as "VAX MIPS." In this case MIPS is just a performance factor for a given machine relative to the performance of a VAX 11/780. If a machine runs some program or set of programs X times faster than a VAX 11/780, it is called an X-MIPS machine. This is based on computer folklore saying that for typical programs a VAX 11/780 performs one million instructions per second. Although this is not true,* the belief is

*Some time ago I ran the Dhrystone benchmark program on VAX 11/780s with different compilers. With Berkeley Unix (4.2) Pascal, the benchmark was translated into 483 instructions executed in 700 microseconds, yielding 0.69 (native) MIPS. With DEC VMS Pascal (V. 2.4), 226 instructions were executed in 543 microseconds, yielding 0.42 (native) MIPS. Interestingly, the version with the lower MIPS rating executed the program faster.

0018-9162/90/1200-065$01.00 © 1990 IEEE

widespread. When VAX MIPS are quoted, it is important to know what programs form the basis for the comparison and what compilers are used for the VAX 11/780. Older Berkeley Unix compilers produced code up to 30 percent slower than VMS compilers, thereby inflating the MIPS rating of other machines.

The MIPS numbers that manufacturers give for their products can be any of the following:

- MIPS numbers with no derivation. This can mean anything, and flippant interpretations such as "meaningless indication of processor speed" are justified.
- Native MIPS, or MIPS in the literal meaning. To interpret this you must know what program the computation was based on and how many instructions are generated per average high-level language statement.
- Peak MIPS. This term sometimes appears in product announcements of new microprocessors. It is largely irrelevant, since it equals the clock frequency for most processors (most can execute at least one instruction in one clock cycle).
- EDN MIPS, Dhrystone MIPS, or similar. This could mean native MIPS, when a particular program is running. More often it means VAX MIPS (see below) with a specific program as the basis for comparison.
- VAX MIPS. A factor relative to the VAX 11/780, which then raises the following questions: What language? What compiler (Unix or VMS) was used for the VAX? What programs have been measured? (Note that DEC uses the term VUP, for VAX unit of performance, in making comparisons relative to the VAX 11/780. These units are based on a set of DEC internal programs, including some floating-point programs.)

In short, Omri Serlin[2] is correct in saying, "There are no accepted industry standards for computing the value of MIPS."

## Benchmarks

Any attempt to make MIPS numbers meaningful (for example, VAX MIPS) comes down to running a representative program or set of programs. Therefore, we can drop the notion of MIPS and just compare the speed for these benchmark programs.

It has been said that the best benchmark is the user's own application. But this is often unrealistic, since it is not always possible to run the application on each machine in question. There are other considerations, too: The program may have been tailored to run optimally on an older machine; original equipment manufacturers must choose a microprocessor for a whole range of applications; journalists want to characterize machine speed independent of a particular application program. Therefore, the next best benchmark (1) is written in a high-level language, making it portable across different machines, (2) is representative for some kind of programming style (for example, systems programming, numerical programming, or commercial programming), (3) can be measured easily, and (4) has wide distribution.

Obviously, some of these requirements are contradictory. The more representative the benchmark program — in terms of similarity to real programs — the more complicated it will be. Thus, measurement becomes more difficult, and results may be available for only a few machines. This explains the popularity of certain benchmark programs that are not complete application programs but still claim to be representative for a given area.

This article concentrates on the most common "stone age" benchmarks (CPU/memory/compiler benchmarks only) — in particular the Whetstone, Dhrystone, and Linpack benchmarks. These are the benchmarks whose results are most often cited in manufacturers' publications and in the trade press. They are better than meaningless MIPS numbers, but readers should know their properties — that is, what they do and don't measure.

Whetstone and Dhrystone are synthetic benchmarks: They were written solely for benchmarking purposes and perform no useful computation. Linpack was distilled out of a real, purposeful program that is now used as a benchmark.

Tables A-D in the sidebar on pages 68-69 give detailed information about the high-level language features used by these benchmarks. Comparing these advantages with the characteristics of the user's own programs shows how meaningful the results of a particular benchmark are for the user's own applications. The tables contain comparable information for all three benchmarks, thereby revealing their differences and similarities.

All percentages in the tables are dynamic percentages, that is, percentages obtained by profiling or, for the language-feature distribution, by adding appropriate counters on the source level and executing the program with counters. Note that for all programs, even those normally used in the Fortran version, the language-feature-related statistics refer to the C version of the benchmarks; this was the version for which the modification was performed. However, since most features are similar in the different languages, numbers for other languages should not differ much. The profiling data has been obtained from the Fortran version (Whetstone, Linpack) or the C version (Dhrystone).

## Whetstone

The Whetstone benchmark was the first program in the literature explicitly designed for benchmarking. Its authors are H.J. Curnow and B.A. Wichmann from the National Physical Laboratory in Great Britain. It was published in 1976, with Algol 60 as the publication language. Today it is used almost exclusively in its Fortran version, with either single precision or double precision for floating-point numbers.

The benchmark owes its name to the Whetstone Algol compiler system. This system was used to collect statistics about the distribution of "Whetstone instructions," instructions of the intermediate language used by this compiler, for a large number of numerical programs. A synthetic program was then designed. It consisted of several modules, each containing statements of some particular type (integer arithmetic, floating-point arithmetic, "if" statements, calls, and so forth) and ending with a statement printing the results. Weights were attached to the different modules (realized as loop bounds for loops around the individual modules' statements) such that the distribution of Whetstone instructions for the synthetic benchmark matched the distribution observed in the program sample. The weights were chosen in such a way that the program executes a multiple of one million of these Whetstone instructions; thus, benchmark results are given as KWIPS (kilo Whetstone instructions per second) or MWIPS (mega Whetstone instructions per second). This way the familiar term "instructions per second" was retained but given a machine-independent meaning.

A problem with Whetstone is that only one officially controlled version exists — the Pascal version issued with the Pascal Evaluation Suite by the British Standards Institution — Quality Assurance (BSI-QAS). Versions in other languages can be registered with BSA-QAS to ensure comparability.

Many Whetstone versions copied informally and used for benchmarking have the print statements removed, apparently with the intention of achieving better timing accuracy. This is contrary to the authors' intentions, since optimizing compilers may then eliminate significant parts of the program. If timing accuracy is a problem, the loop bounds should be increased in such a way that the time spent in the extra statements becomes insignificant.

Users should know that since 1988 there has been a revised (Pascal) version of the benchmark.[3] Changes were made to modules 6 and 8 to adjust the weights and to preclude unintended optimization by compilers. The print statements have been replaced by statements checking the values of the variables used in the computation. According to Wichmann,[3] performance figures for the two versions should be very similar; however, differences of up to 20 percent cannot be ruled out. The Fortran version has not undergone a similar revision, since with the separate compilation model of Fortran the danger of unintended optimization is smaller (though it certainly exists if all parts are compiled in one unit). All Whetstone data in this article is based on the old version; the language-feature statistics are almost identical for both versions.

**Size, procedure profile, and language-feature distribution.** The static length of the Whetstone benchmark (C version) as compiled by the VAX Unix 4.3 BSD C compiler* is 2,117 bytes (measurement loops only). However, because of the program's nature, the length of the individual modules is more important. They are between 40 and 527 bytes long; all except one are less than 256 bytes long. The weights (upper loop bounds) of the individual modules number between 12 and 899.

Table 1 shows the distribution of execution time spent in the subprograms of Whetstone (VAX 11/785, BSD 4.3 Fortran, single precision). The most important, and perhaps surprising, result is that Whetstone spends more than half its time in library subroutines rather than in the compiled user code.

The distribution of language features is shown in Tables A-D in the sidebar on pages 68-69. Some properties of Whetstone are probably typical for most numeric applications (for example, a high number of loop statements); other properties belong exclusively to Whetstone (for example, very few local variables).

*With the Unix 4.3 BSD language systems, it was easier to determine the code size for the C version. The numbers for the Fortran version should be similar.

**Table 1. Procedure profile for Whetstone.***

| Procedure | Percent | What is done there |
|---|---|---|
| Main program | 18.9 | |
| p3 | 14.4 | FP arithmetic |
| p0 | 11.6 | Indexing |
| pa | 1.9 | FP arithmetic |
| User code | 46.8 | |
| | | |
| Trigonometric functions | 21.6 | Sin, cos, atan |
| Other math functions | 31.7 | Exp, log, sqrt |
| Library functions | 53.3 | |
| | | |
| Total | 100 | |

*Because of rounding, all percentages can add up to a number slightly below or above 100.

**Whetstone characteristics.** Some important characteristics should be kept in mind when using Whetstone numbers for performance comparisons.

(1) Whetstone has a high percentage of floating-point data and floating-point operations. This is intentional, since the benchmark is meant to represent numeric programs.

(2) As mentioned above, a high percentage of execution time is spent in mathematical library functions. This property is derived from the statistical data forming the basis of Whetstone; however, it may not be representative for most of today's numerical application programs. Since the speed of these functions (realized as software subroutines or microcode) dominates Whetstone performance to a high degree, manufacturers can be tempted to manipulate the runtime library for Whetstone performance.

(3) As evident from Table D in the sidebar, Whetstone uses very few local variables. When Whetstone was written, the issue of local versus global variables was hardly being discussed in software engineering, not to mention in computer architecture. Because of this unusual lack of local variables, register windows (in the Sparc RISC, for example) or good register allocation algorithms for local variables (say, in the MIPS RISC compilers) make no difference in Whetstone execution times.

(4) Instead of local variables, Whetstone uses a handful of global data (several scalar variables and a four-element array of constant size) repeatedly. Therefore, a compiler in which the most heavily used global variables are allocated in registers (an optimization usually considered of secondary importance) will boost Whetstone performance.

(5) Because of its construction principle (nine small loops), Whetstone has an extremely high code locality. A near 100 percent hit rate can be expected even for fairly small instruction caches. For the same reason, a simple reordering of the source code can significantly alter the execution time in some cases. For example, it has been reported that for the MC68020 with its 256-byte instruction cache, reordering of the source code can boost performance up to 15 percent.

## Linpack

As explained by its author, Jack Dongarra[4] from the University of Tennessee (previously Argonne National Laboratory), Linpack didn't originate as a benchmark. When first published in 1976, it was just a collection (a package, hence the name) of linear algebra subroutines often used in Fortran programs. Dongarra, who collects and publishes Linpack results, has now distilled what was part of a "real life" program into a benchmark that is distributed in various versions.[5]

The program operates on a large matrix

(two-dimensional array); however, the inner subroutines manipulate the matrix as a one-dimensional array, an optimization customary for sophisticated Fortran programming. The matrix size in the version distributed by standard mail servers is 100 × 100 (within a two-dimensional array declared with bounds 200), but versions for larger arrays also exist.

The results are usually reported in millions of floating-point operations per second (Mflops); the number of floating-point operations the program executes can be derived from the array size. This terminology means that the nonfloating-point operations are neglected or, stated another way, that their execution time is included in that of the floating-point operations. When floating-point operations become increasingly faster relative to integer operations, this terminology becomes some-

## Tables covering more than one benchmark

**Table A. Statement distribution in percentages. ***

| Statement | Dhrystone | Whetstone | Linpack/saxpy |
|---|---|---|---|
| Assignment of a variable | 20.4 | 14.4 | - |
| Assignment of a constant | 11.7 | 8.2 | - |
| Assignment of an expression (one operator) | 17.5 | 1.4 | - |
| Assignment of an expression (two operators) | 1.0 | 24.3 | 48.5 |
| Assignment of an expression (three operators) | 1.0 | 1.6 | - |
| Assignment of an expression (>three operators) | - | 6.8 | - |
| One-sided if statement, "then" part executed | 2.9 | 0.5 | - |
| One-sided if statement, "then" part not executed | 3.9 | 0.1 | 2.2 |
| Two-sided if statement, "then" part executed | 4.9 | 4.0 | - |
| Two-sided if statement, "else" part executed | 1.9 | 4.0 | - |
| For statement (evaluation) | 6.8 | 17.3 | 49.3 |
| Goto statement | - | 0.5 | - |
| While/repeat statement (evaluation) | 4.9 | - | - |
| Switch statement | 1.0 | - | - |
| Break statement | 1.0 | - | - |
| Return statement (with expression) | 4.9 | - | - |
| Call statement (user procedure) | 9.7 | 11.9 | - |
| Call statement (user function) | 4.9 | - | |
| Call statement (system procedure) | 1.0 | - | |
| Call statement (system function) | 1.0 | 4.7 | - |
| | 100 | 100 | 100 |

*Because of rounding, all percentages can add up to a number slightly below or above 100.

**Table C. Operand data-type distribution in percentages.**

| Operand Data Type | Dhrystone | Whetstone | Linpack/saxpy |
|---|---|---|---|
| Integer | 57.0 | 55.7 | 67.2 |
| Char | 19.6 | - | - |
| Float/double | - | 44.3 | 32.8 |
| Enumeration | 10.9 | - | - |
| Boolean | 4.2 | - | - |
| Array | 0.8 | - | - |
| String | 2.3 | - | - |
| Pointer | 5.3 | - | - |
| | 100 | 100 | 100 |

what misleading.

For Linpack, it is important to know what version is measured with respect to the following attribute pairs:

• Single/double — Fortran single precision or double precision for the floating-point data.

• Rolled/unrolled — In the unrolled version, loops are optimized at the source level by "loop unrolling": The loop index (say, *i*) is incremented in steps of four, and the loop body contains four groups of statements, for indexes *i*, *i* + 1, *i* + 2, and *i* + 3. This technique saves execution time for most machines and compilers; however, more sophisticated vector machines, where loop unrolling is done by the compiler generating code for vector hardware, usually execute the standard (rolled) version faster.

• Coded BLAS/Fortran BLAS — Linpack relies heavily on a subpackage of basic linear algebra subroutines (BLAS). Coded BLAS (as opposed to Fortran BLAS) means that these subroutines have been rewritten in assembly language. Dongarra has stopped collecting and publishing results for the coded BLAS version and requires that only the Fortran version of these subroutines be used unchanged. However, some results for coded BLAS versions are still cited elsewhere. Computing the execution-time ratio between coded BLAS and Fortran BLAS versions for the same machine offers insights about the Fortran compiler's code optimization quality: For some machines the ratio is 1.2 to 1; for others it can be as high as 2 to 1.

**Table B. Operator distribution in percentages.**

| Operator | Dhrystone | Whetstone | Linpack/saxpy |
|---|---|---|---|
| + (int/char) | 21.0 | 11.9 | 14.1 |
| - (int) | 5.0 | 6.0 | - |
| * (int) | 2.5 | 6.0 | - |
| / (int) | 0.8 | - | - |
| Integer arithmetic | 29.3 | 23.9 | 14.1 |
| + (float/double) | - | 14.9 | 14.1 |
| - (float/double) | - | 2.1 | - |
| * (float/double) | - | 9.3 | 14.1 |
| / (float/double) | - | 4.6 | - |
| Floating-point arithmetic | - | 30.9 | 28.2 |
| <, <= (incl. loop control) | 10.1 | 10.7 | 14.5 |
| Other relational operators | 11.7 | 2.8 | 0.6 |
| Relational | 21.8 | 13.5 | 15.1 |
| Logical | 3.3 | - | 0.2 |
| Indexing (one-dimensional) | 5.9 | 24.5 | 42.3 |
| Indexing (two-dimensional) | 3.4 | - | - |
| Indexing | 9.3 | 24.5 | 42.3 |
| Record selection | 7.6 | - | - |
| Record selection via pointer | 15.1 | - | - |
| Record selection | 22.7 | - | - |
| Address operator (C) | 5.0 | 3.6 | - |
| Indirection operator (C) | 8.4 | 3.6 | - |
| C-specific operators | 13.4 | 7.2 | - |
| Total | 100 | 100 | 100 |

**Table D. Operand locality distribution in percentages.**

| Operand Locality | Dhrystone | Whetstone | Linpack/saxpy |
|---|---|---|---|
| Local | 48.7 | 0.4 | 49.5 |
| Global | 8.3 | 56.3 | - |
| Parameter (value) | 10.6 | 18.6 | 17.0 |
| Parameter (reference) | 6.8 | 1.9 | 24.6 |
| Function result | 2.3 | 1.3 | - |
| Constant | 23.4 | 21.6 | 8.8 |
| | 100 | 100 | 100 |

**Size, procedure profile, and language-feature distribution.** The Linpack data reported here is for the rolled version, single precision, with Fortran BLAS; code sizes have been measured with VAX Unix BSD 4.3 Fortran.

The static code length for all subprograms is 4,537 bytes. The length for individual subprograms varies between 111 and 1,789 bytes; the most heavily used subprogram, saxpy, is 234 bytes long. Data size, in the standard version, is dominated by an array of 100 × 100 real numbers. For 32-bit machines, this means that with single precision, 40 Kbytes are used for data (80 Kbytes with double precision).

Table 2 shows the distribution of execution time in the various subroutines. The most important observation from the table is that more than 75 percent of Linpack's execution time is spent in a 15-line subroutine (called saxpy in the single-precision version and daxpy in the double-precision version). Dongarra[4] reports that on most machines the percentage is even higher (90 percent). Because of this extreme concentration of execution time in the saxpy subroutine, and because of the time-consuming instrumentation method for obtaining the measurements, language-feature distribution has been measured only for the saxpy subroutine (rolled version).

Table A in the sidebar shows that very few statement types (assignment with multiplication and addition, and "for"

Table 2. Procedure profile for Linpack.

| Procedure | Percent | What is done there |
|---|---|---|
| Main program | 0.0 | |
| matgen | 13.8 | |
| sgefa | 6.2 | |
| saxpy | 77.1 | $y[i] = y[i] + a*x[i]$ |
| isamax | 1.6 | |
| Miscellaneous | 1.2 | |
| User code | 100 | |
| Library functions | 0.0 | |

statements) make up the bulk of the subroutine and, therefore, of Linpack itself. The data is mostly reference parameters (array values) or local variables (indexes); there are hardly any global variables.

**Linpack characteristics.** To interpret performance characterizations by Linpack Mflops, it helps to know the benchmark's main characteristics:

- As expected for a numeric benchmark, Linpack has a high percentage of floating-point operations, though only a few are actually used. For example, the program has no floating-point divisions. In striking contrast to Whetstone, no mathematical functions are used at all.
- The execution time is spent almost exclusively in one small function. This means that even a small instruction cache will show a very high hit rate.
- Contrary to the high locality for code, Linpack has a low locality for data. A larger size for the main matrix leads — depending on the cache size — to significantly more cache misses and therefore to a lower Mflops rate. So, in making comparisons, it is important to know whether Linpack Mflops for different machines have been computed using the same array dimensions. Also, Linpack can be highly sensitive to the cache configuration: A different array alignment (201 × 200 instead of 200 × 200 for the global array declaration) can lead to a different mapping of data to cache lines and therefore to a considerably different execution time. The program, as distributed by the standard mail servers, delivers Mflops numbers for two choices of leading dimension, 200 and 201; we can assume that manufacturers report the better number.

## Dhrystone

As the name indicates, Dhrystone was developed much as Whetstone was; it is a synthetic benchmark that I published in 1984. The original language of publication is Ada, although it uses only the Pascal subset of Ada and was intended for easy translation to Pascal and C. It is used mainly in the C version.

The basis for Dhrystone is a literature survey on the distribution of source language features in nonnumeric, system-type programming (operating systems, compilers, editors, and so forth). In addition to the obvious difference in data types (integral versus floating-point), numeric and system-type programs have other differences, too: System programs contain fewer loops, simpler computational statements, and more "if" statements and procedure calls.

Dhrystone consists of 12 procedures included in one measurement loop with 94 statements. During one loop (one Dhrystone), 101 statements (103 in the C version) are executed dynamically. The results are usually given in Dhrystones per second. The program (currently Version 2.1) has been distributed mainly through Usenet, the Unix network; I also make it available on a floppy disk. Rick Richardson has collected and posted results for the Dhrystone benchmark regularly to Usenet (the latest list of results is dated April 29, 1990).

**Size, procedure profile, and language-feature distribution.** The static length of the Dhrystone measurement loop, as compiled by the VAX Unix (BSD 4.3) C compiler, is 1,039 bytes. Table 3 shows the distribution of execution time spent in its subprograms.

The percentage of time spent in string operations is highly language dependent; it drops to 10 percent instead of 16 percent if the Pascal (or Ada) version is used (measurement for Berkeley Unix 4.3 Pascal). On the other hand, the number is higher for newer RISC machines with optimizing compilers, mainly because they spend much less time in procedure calls than the VAX.

Consistent with usage in system-type programming, arithmetic expressions are simpler than in the other benchmarks; there are more "if" statements and fewer loops.

Dhrystone was the first benchmark to explicitly consider the locality of operands: Local variables and parameters are used more often than global variables. This is not only consistent with good software engineering practices but also important for modern CPU architectures (RISC architectures). On older machines with few registers, local variables and parameters are allocated in memory in the same way as global variables; on RISC machines they typically reside in registers. The resulting difference in access time is one of the most important advantages of RISC architectures.

**Dhrystone characteristics.** Familiarity with the benchmark's main characteristics, described below, is important when interpreting Dhrystone performance characterizations.

- As intended, Dhrystone contains no floating-point operations in its measurement loop.
- A considerable percentage of execution time is spent in string functions; this number should have been lower. In extreme cases (MIPS architecture and C compiler), this number goes up to 40 percent.
- Unlike Whetstone, Dhrystone contains hardly any loops within the main measurement loop. Therefore, for microprocessors with small instruction caches (below 1,000 bytes), almost all instruction accesses are cache misses. But as soon as the cache becomes larger than the measurement loop, all instruction accesses are cache hits.
- Only a small amount of global data is manipulated, and the data size cannot be scaled as in Linpack.
- No attempt has been made to thwart optimizing compilers. The goal was for the program to reflect typical programming style; it should be just as optimizable as normal programs. An exception is the optimization of dead-code removal. Since in Version 1 the computation results were not printed or used, optimizing compilers were able to recognize many statements as

dead code and suppress code generation for these statements. In Version 2, this has been corrected.

**Ground rules for Dhrystone number comparisons.** Because of Dhrystone's peculiarities, users should be sure to observe certain ground rules when comparing Dhrystone results. First, the version used should be 2.1; the earlier version, 1.1, leaves too much room for distortion of results by dead-code elimination.

Second, the two modules must be compiled separately, and procedure merging (in-lining) is not allowed for user procedures. ANSI C, however, allows in-lining of library routines (relevant for string routines in the C version of Dhrystone).

Third, when processors are compared, the same programming language must be used on both. For compilers of equal quality, Pascal and Ada numbers can be about 10 percent better because of the string semantics. In C, the length of a string is normally not known at compile time, and the compiler needs — at least for the string comparison statement in Dhrystone — to generate code that checks each byte for the string terminator byte (null byte). With Pascal and Ada the compiler can generate word instructions (usually in-line code) for the string operations.

Therefore, for a meaningful comparison of C-version results, it helps to be able to answer certain questions:

(1) Are the string routines written in machine code?

(2) Are the string routines implemented as in-line code?

(3) Does the compiler use the fact that in the "strcpy" statement the source operand has a fixed length? If it does (legal according to ANSI C), this statement can be compiled in the same way as a record assignment, which can result in considerable savings.

(4) Is a word alignment assumed for the string routines? This is acceptable for the strcpy statement only, not for the "strcmp" statement.

Language systems are allowed to optimize for cases 1 through 3 above, just as they can for programs in general. For processor comparisons, however, it is important that the compilers used apply the same amount of optimization; otherwise, optimization differences may overshadow CPU speed differences. This usually requires an inspection of the generated machine code and the C library routines.

**Table 3. Dhrystone procedure profile.**

| Procedure | Percent | What is done there |
|---|---|---|
| Main program | 18.3 | |
| User procedures | 65.7 | |
| User code | 84.0 | |
| strcpy | 8.0 | String copy (string constant) |
| strcmp | 8.1 | String comparison (string variables) |
| Library functions | 16.1 | |
| Total | 100 | |

## Other benchmarks

In addition to the most often quoted benchmarks explained above, several other programs are used as benchmarks, including

- Livermore Fortran Kernels,
- Stanford Small Programs Benchmark Set,
- EDN benchmarks,
- Sieve of Eratosthenes,
- Rhealstone, and
- SPEC benchmarks.

These range from small, randomly chosen programs such as Sieve, to elaborate benchmark suites such as Livermore Fortran Kernels and SPEC benchmarks.

**Livermore Fortran Kernels.** The Livermore Fortran Kernels, also called the Lawrence Livermore Loops, consist of 24 kernels, or inner loops, of numeric computations from different areas of the physical sciences. The author, F.H. McMahon of Lawrence Livermore National Laboratory, has collected them into a benchmark suite and has added statements for time measurement. The individual loops range from a few lines to about one page of source code. The program is self-measuring and computes Mflops rates for each kernel, for three different vector lengths.

As we might expect, these kernels contain many floating-point computations and a high percentage of array accesses. Several kernels contain vectorizable code; some contain code that is vectorizable if rewritten. (Feo[6] provides a detailed discussion of the Livermore Loops.) McMahon characterizes the representativity of the Livermore Loops as follows:

The net Mflops rate of many Fortran programs and work loads will be in the subrange between the equi-weighted harmonic and arithmetic means, depending on the degree of code parallelism and optimization. The Mflops metric provides a quick measure of the average efficiency of a computer system, since its peak computing rate is well known.

**Stanford Small Programs Benchmark Set.** Concurrent with development of the first RISC systems at Stanford University and the University of California, Berkeley, John Hennessy and Peter Nye at Stanford's Computer Systems Laboratory collected a set of small programs (one page or less of source code for each program). These programs became popular mainly because they were the basis for the first comparisons of RISC and CISC processors. They have now been packed into one C program containing eight integer programs — Permutations, Towers of Hanoi, Eight Queens, Integer Matrix Multiplication, Puzzle, Quicksort, Bubble Sort, and Tree Sort — and two floating-point programs — Floating-point Matrix Multiplication and Fast Fourier Transformation.

Characteristics of the individual programs vary; most contain a high percentage of array accesses. There seems to be no official publication of the source code. The only place I have seen the C code in print is in a manufacturer's performance report.

There is no standardized method for generating an overall figure of merit from the individual execution times. In one version, a driver program assigns weights between 0.5 and 4.44 to the individual execution times. Perhaps a better alternative, used by Sun and MIPS, is to compute the geometric mean of the individual programs' execution times.

Table 4. SPEC benchmark programs.

| Acronym | Short Characterization | Language | Main Data Types |
|---|---|---|---|
| gcc | GNU C compiler | C | Integer |
| espresso | PLA simulator | C | Integer |
| spice 2g6 | Analog circuit simulation | Fortran | Floating point |
| doduc | Monte Carlo simulation | Fortran | Floating point |
| nasa7 | Collection of several numerical "kernels" | Fortran | Floating point |
| li | Lisp interpreter | C | Integer |
| eqntott | Switching-function minimization, mostly sorting | C | Integer |
| matrix300 | Various matrix multiplication algorithms | Fortran | Floating point |
| fpppp | Maxwell equations | Fortran | Floating point |
| tomcatv | Mesh generation, highly vectorizable | Fortran | Floating point |

**EDN benchmarks.** The program collection now known as the EDN benchmarks was developed by a group at Carnegie Mellon University for the Military Computer Family project. *EDN* published it in 1981. Originally, the programs were written in several assembly languages (LSI-11/23, 8086, 68000, and Z8000); the intention was to measure the speed of microprocessors without also measuring the compiler's quality.

A subset of the original benchmarks is often used in a C version:

- Benchmark E: String search
- Benchmark F: Bit test/set/reset
- Benchmark H: Linked list insertion
- Benchmark I: Quicksort
- Benchmark K: Bit matrix transformation

This subset of the EDN benchmarks has been used in Bud Funk's comparison of RISC and CISC processors.[7] There seems to be no standard C version of the EDN benchmarks; the programs are disseminated informally.

**Sieve of Eratosthenes.** One of the most popular programs for benchmarking small PCs is the Sieve of Eratosthenes, sometimes called "Primes." It computes all prime numbers up to a given limit (usually 8,192). The program has some unusual characteristics. For example, 33 percent of the dynamically executed statements are assignments of a constant; only 5 percent are assignments with an expression at the right-hand side. There are no "while" statements and no procedure calls; 50 percent of the statements are loop control evaluations. All operands are integer operands, and 58 percent of them are local variables.

The program is mentioned here not because it can be considered a good benchmark but because, as one author put it, "Sieve performance of one compiler over another has probably sold more compilers for some companies than any other benchmark in history."

**SPEC benchmarks.** Probably the most important current benchmarking effort is SPEC — the systems performance evaluation cooperative effort. It started because benchmarking experts at various companies felt that most previously existing benchmarks (usually small programs) were inadequate. Small benchmarks can no longer be representative for real programs when it comes to testing the memory system, because with the growing size of cache memories and the introduction of on-chip caches for high-end microprocessors, the cache hit ratio comes close to 100 percent for these benchmarks. Furthermore, once a small program becomes popular as a benchmark, compiler writers are inclined (or forced) to "tweak" their compilers into optimizations particularly beneficial to this benchmark — for example, the string optimizations for Dhrystone.

SPEC's goal is to collect, standardize, and distribute large application programs that can be used as benchmarks. This is a nontrivial task, since realistic programs previously used in benchmarking (for example, the Unix utilities "yacc" or "nroff") often require a license and are therefore not freely distributable.

The founding members of SPEC were Apollo, Hewlett-Packard, MIPS, and Sun; subsequently, AT&T, Bull, CDC, Compaq, Data General, DEC, Dupont, Fujitsu, IBM, Intel, Intergraph, Motorola, NCR, Siemens Nixdorf, Silicon Graphics, Solbourne, Stardent, and Unisys became members.

In October 1989, SPEC released its first set of 10 benchmark programs. Table 4 contains only a rough characterization of the programs; J. Uniejewski[8] provides a more detailed discussion. Because a license must be signed, and because of its size (150,000 lines of source code), the SPEC benchmark suite is distributed via magnetic tape only.

Results are given as performance relative to a VAX 11/780 using VMS compilers. Results for several computers of SPEC member companies are contained in the regular *SPEC Newsletter* (see Additional reading and address information). A comprehensive number, the "SPECmark," is defined as the geometric mean of the relative performance of the 10 programs. However, SPEC requires a reporting form that gives, in addition to the raw data, the relative performance for each benchmark program separately. Thus, users can select the subset of performance numbers for which the programming language and/or the application area best matches their applications.

## Non-CPU influences in benchmark performance

In trade journals and advertisements, manufacturers usually credit good benchmark numbers only to the hardware system's speed. With microprocessors, this is reduced even more to the CPU speed. However, the preceding discussion makes it clear that other factors also have an influence — for example, the programming language, the compiler, the runtime library functions, and the memory and cache size.

**Programming-language influence.** Table 5 (numbers from Levy and Clark[9] and my own collection of Dhrystone results) shows the execution time of several programs on the same machine (VAX, 1982 and 1985). Properties of the languages (calling sequence, pointer semantics, and string semantics) obviously influence execution time even if the source programs look similar and produce the same results.

**Compiler influence.** Table 6, taken from the MIPS Performance Brief,[10] gives Dhrystone results (as of January 1990) for the MIPS M/2000 with the MIPS C compiler cc2.0. The table shows how the different levels of optimization influence execution time.

Note that optimization "O4" performs procedure in-lining, an optimization not consistent with the ground rules and included in the report for comparison only. On the other hand, the "strcpy" optimization for Dhrystone is not included in any of the optimization levels for the MIPS C compiler. If it is used, the Dhrystone rate increases considerably.

**Runtime library system.** The role of the runtime library system is often overlooked when benchmark results are compared. As apparent from Table 1, Whetstone spends 40 to 50 percent of the execution time in functions of the mathematical subroutines library. The C version of Dhrystone spends 16 percent of the execution time in the string functions (VAX, Berkeley Unix 4.3 C); with other systems, the percentage can be higher.

Some systems have two flavors of the mathematical floating-point library: The first is guaranteed to comply with the IEEE floating-point standard; the second is faster and may give less accurate results under some circumstances. Customers who must rely on the accuracy of floating-point computations should know which library was used for benchmark measurements.

**Cache size.** It is important to look for the built-in performance boost when the cache size reaches the relevant benchmark size. Depending on the difference between access times for the cache and the main memory, cache size can have a considerable effect.

Table 7 summarizes the code sizes (size of the relevant procedures/inner loops) and data sizes (of the main array) for some popular benchmarks. All sizes have been measured for the VAX 11 with the Unix BSD 4.3 C compiler, with optimization "–O" (code compaction). Of course, the sizes will differ for other architectures and compilers. Typically, RISC architectures lead to larger code sizes, whereas the data size remains the same.

If the cache is smaller than the relevant benchmark, reordering the code can, for some benchmarks and cache configurations, lead to considerable savings in execution time. Such savings have been reported for Whetstone on MC 68020 systems (reordering the source program) as well as for Dhrystone on NS 32532, where just a different linkage order can lead to a difference of up to 5 percent in execution time. It is debatable whether the "good case" or the "bad case" better represents the system's true characteristics. In any event, customers should be aware of these effects and know when the standard order of the code has been changed.

**Table 5. Performance ratio for different languages (larger is better, C = 1): Stanford programs.**

| Program | Bliss | C | Pascal | Ada |
|---|---|---|---|---|
| Search | 1.24 | 1.0 | 0.70 | |
| Sieve | 0.63 | 1.0 | 0.80 | |
| Puzzle | 0.77 | 1.0 | 0.73 | |
| Ackermann | 1.20 | 1.0 | 0.80 | |
| Dhrystone (1.1) | | 1.0 | 1.32 | 1.02 |

**Table 6. Compiler optimization levels in Dhrystones/sec.**

| Optimization Level | V. 1.1 | V. 2.1 |
|---|---|---|
| No opt., no "register" attribute | 30,700 | 31,000 |
| No opt., with "register" attribute | 32,600 | 32,400 |
| Optimization "O," no "register" attribute | 39,700 | 36,700 |
| Optimization "O," with "register" attribute | 39,700 | 36,700 |
| Optimization "O3" | 43,100 | 39,400 |
| Optimization "O4" | 46,700 | 43,200 |

**Table 7. Size in bytes for some popular benchmarks.**

| Program | Code | Data |
|---|---|---|
| Whetstone | ~ 256 | 16 |
| Dhrystone | 1,039 | - |
| Linpack (saxpy) (100×100 version) | 234 | 40,000 |
| Sieve (standard version) | 160 | 8,192 |
| Quicksort (standard version) | 174 | 20,000 |
| Puzzle | 1,642 | 511 |
| Ackermann | 52 | - |

## Small, synthetic benchmarks versus real-life programs

It should be apparent by now that with the advent of on-chip caches and sophisticated optimizing compilers, small benchmarks gradually lose their predictive value. This is why current efforts like SPEC's

## Obtaining benchmark sources via e-mail

Most of the benchmarks discussed in this article can be obtained via electronic mail from several mail servers established at large research institutes.[5] The major mail servers and their electronic mail addresses are shown below. Users can get information about the use of these mail servers by sending electronic mail consisting of the line "send index" to any of the mail servers.

The SPEC benchmarks are available only via magnetic tape.

| | | |
|---|---|---|
| **North America** | | |
| uucp: | uunet!research!netlib | Murray Hill, New Jersey |
| Internet: | netlib@research.att.com | |
| Internet: | netlib@ornl.gov | Oak Ridge, Tennessee |
| **Europe** | | |
| EUNET/uucp: | nac!netlib | Oslo, Norway |
| Internet: | netlib@nac.no | |
| EARN/Bitnet: | netlib%nac.no@norunix.bitnet | |
| X.400: | s=netlib; o=nac; c=no; | |
| **Pacific** | | |
| Internet: | netlib@draci.cs.uow.edu.au | Univ. of Wollongong, NSW, Australia |

activities concentrate on collecting large, real-life programs. Why, then, should this article bother to characterize in detail these "stone age" benchmarks? There are several reasons:

(1) Manufacturers will continue to use them for some time, so the trade press will keep quoting them.

(2) Manufacturers sometimes base their MIPS rating on them. An example is IBM's (unfortunate) decision to base the published (VAX-relative) MIPS numbers for the IBM 6000 workstation on the old 1.1 version of Dhrystone. Subsequently, DEC and Motorola changed the MIPS computation rules for their competing products, also basing their MIPS numbers on Dhrystone 1.1.

(3) For investigating new architectural designs — via simulations, for example — the benchmarks can provide a useful first approximation.

(4) For embedded microprocessors with no standard system software (the SPEC suite requires Unix or an equivalent operating system), nothing else may be available.

(5) We can expect that larger benchmarks will not be completely free of distortions from unforeseen effects either. Experience gained with smaller benchmarks can help us be aware of such effects. For example, it won't be as easy to tweak compilers for the SPEC benchmarks as it is for the small benchmarks; but if it happens, it also will be harder to detect.

Advice for users looking at benchmark numbers to characterize machine performance should begin with a warning not to trust MIPS numbers unless their derivation is clearly explained. Here are some other things to watch for:

- Check whether Mflops numbers relate to a standard benchmark. Does this benchmark match your applications?
- Know the properties of the benchmarks whose results are advertised.
- Be sure you know all the relevant facts about your system and the manufacturer's benchmarking system. For hardware this includes clock frequency, memory latency, and cache size; for software it includes programming language, code size, data size, compiler version, compiler options, and runtime library.
- Check benchmark code listings to make sure apples are compared with apples and that no illegal optimizations are applied.
- Ask for a well-written performance report. Good companies provide all relevant details. ■

## References

1. D.A. Patterson, "Reduced Instruction-Set Computers," *Comm. ACM*, Vol. 28, No. 1, Jan. 1985, pp. 8-21.
2. O. Serlin, "MIPS, Dhrystones, and Other Tales," *Datamation*, June 1986, pp. 112-118.
3. B.A. Wichmann, "Validation Code for the Whetstone Benchmark," Tech. Report NPL-DITC 107/88, National Physical Laboratory, Teddington, UK, Mar. 1988.
4. J.J. Dongarra, "The Linpack Benchmark: An Explanation," in *Evaluating Supercomputers*, Aad J. Van der Steen, ed., Chapman and Hall, London, 1990, pp. 1-21.
5. J. Dongarra and E. Grosse, "Distribution of Mathematical Software via Electronic Mail," *Comm. ACM*, Vol. 30, No. 5, May 1987, pp. 403-407.
6. J.T. Feo, "An Analysis of the Computational and Parallel Complexity of the Livermore Loops," *Parallel Computing*, Vol. 7, No. 2, June 1988, pp. 163-185.
7. B. Funk, "RISC and CISC Benchmarks and Insights," *Unisys World*, Jan. 1989, pp. 11-14.
8. J. Uniejewski, "Characterizing System Performance Using Application-Level Benchmarks," *Proc. Buscon*, Sept. 1989, pp. 159-167. Partial publication in *SPEC Newsletter*, Vol. 2, No. 3, Summer 1990, pp. 3-4.
9. H. Levy and D.W. Clark, "On the Use of Benchmarks for Measuring System Performance," *Computer Architecture News*, Vol. 10, No. 6, Dec. 1982, pp. 5-8.
10. MIPS Computer Systems, Inc., *Performance Brief, CPU Benchmarks*, Issue 3.9, Jan. 1990, p. 35.

## Additional reading and address information

Following are the main reference sources for each of the benchmarks discussed in this article, together with a short characterization. A contact person is identified for each of the major benchmarks so that readers can get additional information. For information about access to the benchmark sources via electronic mail, see the sidebar "Obtaining benchmark sources via e-mail."

### Whetstone

Curnow, H.J., and B.A. Wichmann, "A Synthetic Benchmark," *The Computer J.*, Vol. 19, No. 1, 1976, pp. 43-49. Original publication, explanation of the benchmark design, program (Algol 60) in the appendix.

Wichmann, B.A., "Validation Code for the Whetstone Benchmark," see Reference 3. Discussion of comments made to the original program, explanation of the revised version. Paper contains a program listing of the revised version, in Pascal, including checks for correct execution.

Contact: Brian A. Wichmann, National Physical Laboratory, Teddington, Middlesex, England TW11 OLW; phone 44 (81) 943-6976, fax 44 (81) 977-7091, Internet baw@seg.npl.co.uk.

Registration of other versions: J.B. Souter, Benchmark Registration, BSI-QAS, PO Box 375, Milton Keynes, Great Britain MK14 6LL.

### Linpack

Dongarra, J.J., et al., *Linpack Users' Guide*, SIAM Publications, Philadelphia, Pa., 1976. Original publication (not yet as a benchmark), contains the benchmark program as an appendix.

Dongarra, J.J., "Performance of Various Computers Using Standard Equations Software in a Fortran Environment," *Computer Architecture News*, Vol. 18, No. 1, Mar. 1990, pp. 17-31. Latest published version of the regularly maintained list of Linpack results, rules for Linpack measurements.

Dongarra, J.J., "The Linpack Benchmark: An Explanation," see Reference 4. Explanation of Linpack, guide to interpretation of Linpack results.

Contact: Jack J. Dongarra, Computer Science Dept., Univ. of Tennessee, Knoxville, TN 37996-1301; phone (615) 974-8295, fax (615) 974-8296, Internet dongarra@cs.utk.edu.

### Dhrystone

Weicker, R.P., "Dhrystone: A Synthetic Systems Programming Benchmark," *Comm. ACM*, Vol. 27, No. 10, Oct. 1984, pp. 1,013-1,030. Original publication, literature survey on the use of programming language features, base statistics and benchmark program in Ada.

Weicker, R.P., "Dhrystone Benchmark: Rationale for Version 2 and Measurement Rules," *SIGPlan Notices*, Vol. 23, No. 8, Aug. 1988, pp. 49-62. Version 2.0 of Dhrystone (in C), measurement rules. For the Ada version, a similar article appeared in *Ada Letters*, Vol. 9, No. 5, July 1989, pp. 60-82.

Weicker, R.P., "Understanding Variations in Dhrystone Performance," *Microprocessor Report*, Vol. 3, No. 5, May 1989, pp. 16-17. What customers should know when C-version results of Dhrystone are compared; reiteration of measurement rules.

Contact: Reinhold P. Weicker, Siemens Nixdorf Information Systems, STM OS 32, Otto-Hahn-Ring 6, W-8000 München 83, Germany; phone 49 (89) 636-42436, fax 49 (89) 636-48008, Internet: weicker@ztivax.siemens.com; Eunet: weicker%ztivax.uucp@unido.uucp.

Collection of results: Rick Richardson, PC Research, Inc., 94 Apple Orchard Dr., Tinton Falls, NJ 07724; phone (201) 389-8963, e-mail (UUCP) ...!uunet!pcrat!rick.

### Livermore Fortran Kernels

Feo, J.T., "An Analysis of the Computational and Parallel Complexity of the Livermore Loops," see Reference 6. Analysis of the Livermore Fortran Kernels with respect to the achievable parallelism.

McMahon, F.H., "The Livermore Fortran Kernels: A Computer Test of the Numerical Performance Range," Tech. Report UCRL-53745, Lawrence Livermore National Laboratory, Livermore, Calif., Dec. 1986, p. 179. Original publication of the benchmark with sample results.

McMahon, F.H., "The Livermore Fortran Kernels Test of the Numerical Performance Range," in *Performance Evaluation of Supercomputers*, J.L. Martin, ed., North Holland, Amsterdam, 1988, pp. 143-186. Reprint of main part of the original publication.

Contact: Frank H. McMahon, Lawrence Livermore National Laboratory, L-35, PO Box 808, Livermore, CA 94550; phone (415) 422-1647, Internet mcmahon@ocfmail.ocf.llnl.gov.

### Stanford Small Programs Benchmark Set

Appendix 2 — Stanford Composite Source Code, Appendix to "Performance Report 68020/68030 32-bit Microprocessors," Motorola, Inc., BR705/D, 1988, pp. A2-1 — A2-15. This is the only place I have seen this benchmark in print; it is normally distributed via informal channels.

### EDN benchmarks

Grappel, R.D., and J.E. Hemenway, "A Tale of Four μPs: Benchmarks Quantify Performance," *EDN*, Apr. 1, 1981, pp. 179-265. Original publication with benchmarks described in assembler (code listings for LSI-11/23, 8086, 68000, and Z8000).

Patstone, W., "16-bit-μP Benchmarks — An Update with Explanations," *EDN*, Sept. 16, 1981, pp. 169-203. Discussion of results, updated code listings (assembler).

### Sieve

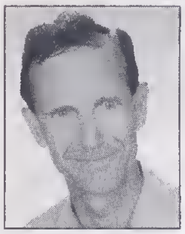
Gilbreath, J., and G. Gilbreath, "Eratosthenes Revisited," *Byte*, Jan. 1983, pp. 283-326. Program listings in Pascal, C, Forth, Fortran IV, Basic, Cobol, Ada, and Modula-2.

### SPEC benchmarks

"Benchmark Results," *SPEC Newsletter*, Vol. 1, No. 1, Fall 1989, pp. 1-15. First published list of results, in the report form required by SPEC.

Uniejewski, J., "Characterizing System Performance Using Application-Level Benchmarks," see Reference 8. This paper includes a short characterization of each SPEC benchmark program.

Contact: SPEC — System Performance Evaluation Cooperative (Kim Shanley, Secretary), c/o Waterside Associates, 39150 Paseo Padre Pkwy., Suite 350, Fremont, CA 94538; phone (415) 792-2901, fax (415) 792-4748, Internet shanley@cup.portal.com.



**Reinhold P. Weicker** is a senior staff engineer with Siemens Nixdorf Information Systems AG in Munich, Germany. His research interests include performance evaluation with benchmarks and its relation to CPU architecture and compiler code generation. He wrote the often-used Dhrystone benchmark while working on the CPU architecture team for the i80960 microprocessor. Previously, he performed research in theoretical computer science at the University of Hamburg, Germany, and was a visiting assistant professor at Pennsylvania State University.

Weicker received a diploma degree in mathematics and a PhD in computer science from the University of Erlangen-Nürnberg. He is a member of the IEEE Computer Society, the ACM, and the Gesellschaft für Informatik.

The author can be contacted at Siemens Nixdorf Information Systems AG, Otto-Hahn-Ring 6, W-8000 München 83, Germany; Internet: weicker@ztivax.siemens.com; Eunet: weicker%ztivax.uucp@unido.uucp.

Editor: Fletcher J. Buckley, 103 Wexford Dr., Cherry Hill, NJ 08003, phone (609) 866-6350, fax (609) 866-7753, Compmail II, f.buckley

## Information technology standardization is key area of interest to IEE

*Alasdair Kemp, Institution of Electrical Engineers*

Standards-making involves a number of agencies and processes in the United Kingdom, as it does elsewhere. Interleaving takes place, and different organizations assume various roles.

In the UK specifically, the British Standards Institution, the national standards authority, counts on volunteer participation in its technical committees to fulfill its mission of developing standards. The BSI rarely delegates its responsibility to other bodies.

Similarly, the European Community has its own standards authorities: Cen and Cenelec. In addition, Directorate General XIII of the European Commission is responsible for developing EC information technology.

In contrast to the American National Standards Institute in the US, the BSI takes on the task of providing secretariat support for virtually all national standards-making in the UK — rather than delegating the task to other organizations and then approving the resultant standards. The actual drafting of standards depends on the volunteer participation of delegates from professional bodies, trade associations, and — sometimes — government departments.

Where information technology is concerned, the BSI has been found wanting. To remedy this, Project DISC, a self-financing independent organization, hopes to provide a new industry-funded body for IT standards.

The nature of IT is driving standards towards increasing internationality. This is particularly true with respect to the approaching introduction of the single European market in 1992. The European Commission recently issued a consultative document (Green Paper) containing proposals for changes to the organization of standards-making in Europe. It puts forward the view that technical standards are vital to the success of the single market.

It is widely believed that national standards should be the same as international standards promulgated through the IEC, the ISO, and the IEC/ISO Joint Technical Committee. This process of harmonization is both more essential and more practical in IT than in other areas and is often achieved by *force majeure*. The speed of development and the short life cycle of IT products affect other aspects of standards development and consequently provide the underlying bases for Project DISC.

*Wiring Regulations*, the definitive standards for the electrical industry produced and published by the Institution of Electrical Engineers and widely used throughout the British Commonwealth, constitutes one of the exceptions to the general rule of standards-making in the UK. A small committee of IEE members, called a working party, is considering the role of the institution regarding standards-making for all areas in which members are professionally involved. This approach was embraced in response to a suggestion that the IEE adopt a wider and more active role in IT standardization.

**The nature of information technology is driving standards towards increasing internationality. This is particularly true with respect to the approaching introduction of the single European market in 1992.**

**IT standards and the IEE.** The IEE is deeply concerned with the entire IT field. Its long-standing IT Standards Subcommittee operates under the direction of the Computing and Control Division and, in addition to providing the focal point for IEE IT standards activity, undertakes joint activities with other interested bodies.

The subcommittee is also responsible for informing interested individuals of standards and the benefits of their use, for identifying gaps in standards provision (the availability of a standard for a particular purpose/set of circumstances), and for taking initiatives to fill voids.

In part, this subcommittee operates through working parties, some of which have been concerned with documents that are specified below. Working parties currently function in the areas of computer systems architecture, open systems in manufacturing, and dissemination of information technology standards. An IEE/BCS Joint Working Party on Software Engineering Standards is also active.

The objective of the Computing Systems Architecture Working Party is to produce a general model that can provide a framework to relate existing standards and projects. The idea is to detect gaps. This is a large undertaking, but progress is being made.

The purpose of the Open Systems in Manufacturing Working Party is to promote awareness of appropriate standards for manufacturing information formats and disseminate manufacturing information around industry. This working party focuses on small- and medium-sized organizations with up to 500 employees. The WP is taking a general approach, starting with the documentation of initial inquiries, proceeding with design and manufacturing processes, and concluding with post-scale and service documentation.

The need for interconnectivity of equipment and transfer of data and software — that is, open systems — has provided further motivation for interest in standards. Despite this, some recent reports have shown that a remarkable ignorance of open systems concepts exists, even on the part of those who reasonably might be expected to be knowledgeable. We at the IEE hope that the working party will produce documentation that will help IT newcomers avoid piecemeal procurement of systems and software, a development that would inevitably lead to incompatibility and unnecessary cost.

The aforementioned IEE/BCS JWP

deals with software engineering standards and has members from both organizations. In the past year, this JWP has been largely concerned with software safety standards. It has also been considering the evaluation of computer-aided software engineering tools, with particular focus on whether the nature of the use of the tools implies achievement of general engineering quality standards (such as with BS 5750/ISO 9000, *Quality Systems*).

The IEE has taken a keen interest in the work of IEEE P1209 (*Recommended Practice for the Evaluation and Use of CASE Tools*), is staying fully informed of this body of work, and has offered to host the summer 1991 IEEE P1209 meeting. The possibility of closer collaboration between the JWP and IST/15, the BSI Technical Committee responsible for software engineering standards, is being explored.

I will discuss the Working Party on Dissemination of IT Standards later in this article.

Membership of working parties is limited to seven experts who are mainly — but not necessarily — institution members. The working parties meet every few weeks. In a number of cases, members of larger consultative committees comment on drafts of documents under preparation. The role of the members of these consultative committees appears similar to that of IEEE balloting committee members, although they are procedurally less formal. The consultative committees meet infrequently, if at all.

Two units, the Information Engineering Committee and the Safety Critical Systems Committee, deal with general IT policy matters for the institution. They enjoy a good deal of cross-representation, and both report to the IEE Public Affairs Board.

**Standards services to members.** The IEE Technical Information Unit staff members strive to keep abreast of developments in IT standards, scanning relevant journals and accessing databases to carry out customized searches for members at cost-effective rates. Of the 25 databases from around the world dedicated to standards information, the staff members mainly access two: the BSI Standardline and IHS International Standards and Specifications.

The BSI Standardline contains bibliographic references to all current British standards, including drafts for development and drafts for public comment from January 1986 to the present. The IHS database contains bibliographic references to industry standards and to military and federal specifications and standards covering all aspects of engineering and related disciplines. It contains information from more than 70 US, foreign (for example, AFNOR in France), and international (for example, ISO) standardizing bodies.

The IEE/BCS library, housed in the institution's headquarters, features a collection of standards, including IEEE, BSI, IEC, and ECMA standards, plus CCIR and CCITT recommendations. Most are available for reference only.

**Improving access to standards information.** Unfortunately, few of the databases cover standards under development. For some time, there has been a perceived need to improve access to information about standards relating to IT. The need is exacerbated by the prevalence of de facto standards as opposed to standards produced by recognized standards-making bodies and by the fact that it can be more important to know about emerging standards than about those that have been formally adopted.

The Gavel Consortium, a Europewide group of consultants, recently reviewed European needs for information about standards. It concluded that, while there was generally insufficient interest in improving sources of information about standards, a new, improved information service related specifically to IT might be useful and economically viable.

HITS (Database) Ltd. was established in 1989 as an independent commercial organization to meet the need for improved information about standards. It intends to have two products, a handbook of information technology standards and a computer database that may be offered on-line and via CD-ROM.

The IEE has had considerable influence on the developments, although it has only been involved in a monitoring role. It has an observer on the HITS board and a representative on its technical committee, and it has established the Dissemination of IT Standards Working Party to review progress. As this was being written, it appeared the IEE might move toward collaboration with one or more private organizations that have established databases, in hopes of helping them provide the kind of information and service considered necessary.

**Standards development.** For its part, the IEE first participates in the work of outside bodies. It has been represented by members and occasionally the secretariat on BSI technical committees, and other national and international committees and working parties.

Second, IEE members comment on drafts, proposals, and revisions. A current example is the Draft Interim Defence Standards for Safety-Related Software proposed by the UK Ministry of Defence. Copies of the institution's comments on this draft (PAB(S) 201) may be obtained from the IEE Public Affairs Board Secretariat. The draft has generated considerable interest, mostly because of criticism of the overemphasis on formal methods.

In addition to commenting on the proposed MOD Interim Draft Defence Standards 00-55 and 00-56, the IEE published a report in October 1989 entitled *Software in Safety-Related Systems* concerned with achieving assurance of safety in systems incorporating software. It was compiled by a joint IEE/BCS project management team and is available from the IEE Publication Sales office.

The IEE monitors IEEE activities, es-

## Glossary

| | |
|---|---|
| ANSI | American National Standards Institute |
| AQAP | Allied Quality Assurance Publications (NATO) |
| BCS | British Computer Society |
| BSI | British Standards Institution |
| CCIR | The International Radio Consultative Committee |
| CCITT | International Consultative Committee for Telephone and Telegraph |
| Cen | The European Committee for Standardization |
| Cenelec | The European Committee for Electrotechnical Standardization |
| ECMA | European Computer Manufacturers Association |
| HIT | Hierarchical Interconnection Technology |
| HITS | Handbook of information technology standards |
| IEC | International Electrotechnical Commission |
| IEE | Institution of Electrical Engineers |
| IHS | Information Handling Services |
| ISO | International Organization for Standardization |
| IT | Information technology |
| JWP | Joint working party |
| MOD | UK Ministry of Defence |
| Project DISC | Project for Delivering Information Solutions to Customers |

pecially where they relate directly to the activities of IEE working parties. It also contributes to the IEEE computer standardization process and sends delegates to the US to participate in meetings. This was particularly the case during the formative IEEE P896 process when the UK provided the chair and a number of members, and carried out most of the editorial work.

Many IEEE committees have active UK members, and there are other more or less formal contacts. As this article was being written, comments were being prepared on the proposed standard for safety plans and, as noted above, the IEE is looking forward to hosting the 1991 P1209 working group meeting on the evaluation and selection of CASE tools.

Third, the institution may take an existing industry or company standard and, with appropriate modification where necessary, make it available to a wider audience.

Occasionally, the IEE will recognize the need for a completely new standard. Hierarchical Interconnection Technology is the most recent instance of this nature. HIT is being considered as a British standard and eventually will be proposed as an international standard. The basic concept of HIT was to allow physical partitioning of elements that would otherwise be put on a single printed circuit board so that the individual elements would be cheap enough to replace in the field. From a standards-making viewpoint, the project involved the cooperation of a number of competing firms, support from the UK government, independent evaluation, and secretariat support from the institution.

More commonly, the need is for guidelines. Four publications of interest to software engineers are now available. *The Software Inspection Handbook* is a guide intended to help with the review of software development in a relatively early part of the development life cycle.

*Guidelines for the Documentation of Computer Software for Real-Time and Interactive Systems* is a second edition, its changed title reflecting the wider range of systems with which it now deals. Much of the practice suggested is equally useful in business and commercial data-processing systems. Additional sections, for example, relate to feasibility studies. Other changes reflect suggestions that users of the first edition made.

*Software Quality Assurance: Model Procedures* contains procedures that have been tested in use and have satisfied the requirements of AQAP-1 and AQAP-13. The procedures described are based on top-down functional decomposition, with programming in a procedural language such as C.

*Guidelines for Assuring Testability* is designed to show how costs and delays can be avoided if the need for testing is taken into account at the earliest stages of a project life cycle, as well as at all subsequent stages. It also gives guidance on how testability can be achieved. In addition to discussing general principles, specific sections deal with software, electronics, and electromechanical products and systems.

**Conclusion.** Compared to the ANSI, the BSI takes a more centralized role relative to IT standards. Although the IEE sees the need for involvement in IT standards and has the opportunity for involvement with these standards, the institution does not currently provide secretariat support for this activity. Resources available within the IEE for dealing with IT standards are limited, and necessity demands the institution be selective in what it seeks to achieve.

IEE activity is also limited by the availability of expert members willing to participate in specific areas. Nonetheless, IT standardization persists as one of the institution's major areas of interest and continues to be important in relation to achieving its overall objectives.

Master of Software Engineering at Carnegie Mellon University

Reply to
MSE Admissions Coordinator
CMU / SEI
Dept. D
Pittsburgh, PA 15213-3890

(412) 268-7713

Reader Service Number 4

**Department of Computer Science**
**University of Minnesota**

The Department of Computer Science at the University of Minnesota invites applications for regular faculty positions in the areas of theory, applied graphics, compilers, and systems. Highest priority will be given to appointments at the full and associate professor levels, although applications will be accepted at all faculty ranks. Candidates for full and associate professorships are required to have established a record of effectiveness in teaching and a reputation in scholarly research; candidates for assistant professorships are required to have demonstrated teaching and research abilities. Minimum academic qualifications for all ranks is an earned doctorate. Minimum teaching and research experience beyond the Ph.D. is 3 years for associate professor and 6 years for full professor.

Non-regular positions as teaching specialists, lecturers, and post-doctoral associates may also be available in all areas of computer science.

Minnesota is a major center of the computer industry. The Computer Science Department currently has 28 full-time faculty members. The activities of the department are augmented by the activities of the Minnesota Supercomputer Institute, the Microelectronics and Information Science Center, the Institute for Mathematics and its Applications, and the Charles Babbage Institute for the History of Information Processing.

Applications, including the names of at least four references, should be sent to the Chair of the Faculty Search Committee, Department of Computer Science, 4-192 EE/CSci Building, University of Minnesota, 200 Union St., S.E., Minneapolis, Minnesota 55455. The closing date for receipt of applications is February 28, 1991.

The University of Minnesota is an equal opportunity educator and employer and specifically invites and encourages applications from women and minorities and persons with disabilities.

1951-1991
40 YEARS OF SERVICE
IEEE COMPUTER SOCIETY

# IEEE COMPUTER SOCIETY Membership / Subscription Application

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

## BENEFITS

**Computer**
You automatically receive *Computer* with membership. Written, reviewed, and refereed by experts, it features survey and tutorial articles covering the entire computer field, and departments such as new products, product reviews, standards, and a reader forum called "The Open Channel." (monthly).

**Technical Committees**
Participate in one or more of our 33 technical committees — networks of professionals with common interests in specialty areas within computer hardware, software, and applications.

**Standards Working Groups**
Participate in the development of the more than 100 standards projects currently sponsored by the society in such diverse areas as software engineering, local area networks, microprocessor buses, design automation, programming languages, and standards definitions.

**Computer Society Press Books and Videos**
Receive discounts of up to 50% on over 700 titles covering a broad spectrum of computer science topics such as networking, communications, advanced systems, image processing, security, artificial intelligence, and visualization. Over 120 new products are published annually.

**Conferences and Tutorials**
Choose from more than 100 conferences annually, ranging from large industry-oriented conferences replete with exhibits to small, highly interactive workshops. Members receive special low rates.

## Schedule of Fees

**To join: see item 1, 2, or 3.**
**To subscribe: see item 4.**

Membership dues and periodical subscriptions are annualized to, and expire on, December 31. Pay full- or half-year rate depending on date of receipt by the Computer Society as indicated below.

| | Half Year<br>Mar 1-Aug 31 | Full Year<br>Sept 1-Feb 28 |
|---|---|---|
| **1** I don't belong to the IEEE and I want to join just the Computer Society | ☐ $27.00 | ☐ $ 54.00 |
| **2** I don't belong to the IEEE and I want to join both the Computer Society and the IEEE* | | |
| I reside in Region 1-6 (United States) | ☐ $52.50 | ☐ $105.00 |
| I reside in Region 7 (Canada) | ☐ $48.50 | ☐ $ 97.00 |
| I reside in Region 8 (Europe, Africa, or the Middle East) | ☐ $48.00 | ☐ $ 96.00 |
| I reside in Region 9 (Latin America) | ☐ $44.50 | ☐ $ 89.00 |
| I reside in Region 10 (Asia and Pacific) | ☐ $43.50 | ☐ $ 87.00 |

*ACM members who join both IEEE and the Computer Society may deduct $5 off the full-year rate; $2.50 off the half-year rate.

| | Half Year | Full Year |
|---|---|---|
| **3** I already belong to the IEEE and I want to join the Computer Society | ☐ $11.00 | ☐ $ 22.00 |

IEEE Member Number ____________

**4** OPTIONAL PERIODICALS for new or current members

| | issues per year | | |
|---|---|---|---|
| *IEEE Computer Graphics and Applications* | 6 | ☐ $12.00 | ☐ $ 24.00 |
| *IEEE Design and Test* | 4 | ☐ $11.00 | ☐ $ 22.00 |
| *IEEE Expert* | 6 | ☐ $10.00 | ☐ $ 20.00 |
| *IEEE Micro* | 6 | ☐ $10.50 | ☐ $ 21.00 |
| *IEEE Software* | 6 | ☐ $12.50 | ☐ $ 25.00 |
| *Transactions on:* | | | |
| *Computers* | 12 | ☐ $12.00 | ☐ $ 24.00 |
| *Knowledge and Data Engineering* | 4 | ☐ $ 7.50 | ☐ $ 15.00 |
| *Parallel and Distributed Systems* | 4 | ☐ $ 7.00 | ☐ $ 14.00 |
| *Pattern Analysis and Machine Intelligence* | 12 | ☐ $12.00 | ☐ $ 24.00 |
| *Software Engineering* | 12 | ☐ $11.00 | ☐ $ 22.00 |

Total amount remitted with this application $ ____________

DC residents add sales tax to optional periodicals.

☐ Checks accepted in Belgian, British, German, Swiss, Japanese, or US currencies. Checks must be drawn on a bank in the country of origin of the currency.

PRICES EXPIRE 12/31/91

☐ Visa ☐ Master Card ☐ American Express

Charge Card Number (Minimum Charge $10.) — Exp. Date Mo. Yr.

I hereby make application for Computer Society membership and, if elected, will be governed by IEEE's and the society's constitutions, bylaws, and statements of policies and procedures.

**MAILING ADDRESS**

Full signature — Date

☐ Male ☐ Female

First name — Middle initial(s) — Last name — Date of birth

Street address — City — State/Country — Zip

**EDUCATION** (highest level completed)

Course — Degrees received — Date

Name of educational institution

**OCCUPATION**

Title or Position

Firm name

Firm address

City — State/Country — Zip

**ENDORSER** (an IEEE member, Senior Member, or Fellow who knows you professionally)

Endorser's signature

Name (print in full) — IEEE Member No.

Street address

City — State/Country — Zip

**Return to:** IEEE Computer Society, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720–1264 USA.
**Residents of Europe mail to:** IEEE Computer Society, 13, Avenue de l'Aquilon, B-1200, Brussels, BELGIUM.
**Asian/Pacific residents mail to:** IEEE Computer Society, Ooshima Building, 2-19-1 Minami-Aoyama, Minato-ku, Tokyo 107 JAPAN.

PC1290

## ADVERTISER INDEX

## FOR DISPLAY ADVERTISING INFORMATION, CONTACT:

**Northern California and Pacific Northwest:** John D. Vance & Associates, Inc., 4030 Moorpark Ave., Suite 116, San Jose, CA 95117; (408) 741-0354.

**Southern California and Mountain States:** Richard C. Faust Co., 24050 Madison St., Suite 101, Torrance, CA 90505; (213) 373-9604.

**Midwest:** The Kingwill Company, 4433 W. Touhy Ave., #540, Lincolnwood, IL 60646; (708) 675-5755.

**East Coast:** Atlantic Representative Group, 349 Maple Pl., Keyport, NJ 07735; (908) 739-1444.

**New England:** Arpin Associates, P.O. Box 6444, Holliston, MA 01746; (508) 429-8907.

**Europe:** Heinz J. Görgens, Parkstrasse 8a, D-4054 Nettetal 1 - Hinsbeck (F.R.G.); phone: (0 21 53) 8 99 88; fax: ( 0 21 53) 8 99 89.

**Southwest/Southeast:** Heidi Rex, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380.

For production information, conference, and classified advertising, contact Heidi Rex or Marian Tibayan.

*COMPUTER*, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264; phone (714) 821-8380; fax (714) 821-4010.

## PRODUCT INDEX

| | RS# | PG# |
|---|---|---|
| Advanced Micro Devices | 120 | 98 |
| Alsys, Inc. | 7, 43 | 7, 97 |
| Analog Devices | 121 | 98 |
| Artisoft Inc. | 24-26 | 88 |
| AST Research | 31 | 93 |
| CACI Products Company | — | 1 |
| Cirrus Logic | 122 | 98 |
| Dariana Technology Group Inc. | 27 | 90 |
| Datacap Inc. | 22 | 85 |
| Decision Data | 38 | 95 |
| Delrina Technology Inc. | 28 | 90 |
| Digitalk | 36 | 95 |
| Doctor Design | 135 | 99 |
| DP-Tek, Inc. | 21 | 85 |
| Hewlett-Packard | 29, 44 | 92, 97 |
| Hitachi | 123 | 98 |
| HNC | 1 | 4-5 |
| ICS Electronics | 136 | 99 |
| Ingres Corp. | 37 | 95 |
| Integrated Device Technology | 124 | 98 |
| International CMOS Technology | 125 | 98 |
| International Computers | 34 | 94 |
| Klever Computers | 137 | 99 |
| Maxell Corp. | 35 | 95 |
| Meiko World | 33 | 94 |
| Mercury | 138 | 99 |
| Mesa Electronics | 139 | 99 |
| Micro Focus | 42 | 96 |
| Micro Linear | 126 | 98 |
| MNC International | 140 | 99 |
| Motorola | 127 | 98 |
| Newer Technology | 141 | 99 |
| NMB Technologies | 128 | 98 |
| Oakland Group Inc. | 2 | C.III |
| Open Software Foundation | 40 | 96 |
| Precision Inc. | 23 | 87 |
| Reply Corp. | 32 | 93 |
| Ricoh | 39 | 96 |
| SBE | 142 | 99 |
| SGS-Thomson | 129 | 98 |
| Software Engineering Institute | 4 | 78 |
| Stac Electronics | 45 | 97 |
| StarSys, Inc. | 3 | 12 |
| Tetra Systems | 41 | 96 |
| Texas Instruments | 30 | 93 |
| Themis | 143 | 99 |
| Twinhead | 130 | 98 |
| Win Systems | 144 | 99 |
| Zeos International | 145 | 99 |

COMPUTER

**Reader Service Card**

**December 1990 issue** (Card void after June 1991)
Print address below.

Name ______________________

Title ______________________

Company ______________________

Address ______________________

City ______________________

State/ZIP ______________________

Country ______________ Phone ______________

**Please send** (Circle those you want)

**201** Publications catalog
**202** Membership information
**203** Student membership application
**204** Application for senior membership

**Reader interest**
(Circle what you liked; add comments on the back)

**A** Parallel Languages
**B** Compiling Scientific Code
**C** Parallel Computation
**D** Engineering Systems
**E** Benchmark Overview
**F** President's Message
**G** EIC Message
**H** Standards
**J** Update
**K** CS News
**L** Product Reviews
**M** New Products
**N** ICs/Microsystems
**P** Conferences
**Q** Call for Papers
**R** Calendar
**S** Book Reviews
**T** CS Magazines
**U** Annual Index

**Product information**
(Circle numbers for products and advertisements on which you want more information)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 21 | 41 | 61 | 81 | 101 | 121 | 141 | 161 | 181 |
| 2 | 22 | 42 | 62 | 82 | 102 | 122 | 142 | 162 | 182 |
| 3 | 23 | 43 | 63 | 83 | 103 | 123 | 143 | 163 | 183 |
| 4 | 24 | 44 | 64 | 84 | 104 | 124 | 144 | 164 | 184 |
| 5 | 25 | 45 | 65 | 85 | 105 | 125 | 145 | 165 | 185 |
| 6 | 26 | 46 | 66 | 86 | 106 | 126 | 146 | 166 | 186 |
| 7 | 27 | 47 | 67 | 87 | 107 | 127 | 147 | 167 | 187 |
| 8 | 28 | 48 | 68 | 88 | 108 | 128 | 148 | 168 | 188 |
| 9 | 29 | 49 | 69 | 89 | 109 | 129 | 149 | 169 | 189 |
| 10 | 30 | 50 | 70 | 90 | 110 | 130 | 150 | 170 | 190 |
| 11 | 31 | 51 | 71 | 91 | 111 | 131 | 151 | 171 | 191 |
| 12 | 32 | 52 | 72 | 92 | 112 | 132 | 152 | 172 | 192 |
| 13 | 33 | 53 | 73 | 93 | 113 | 133 | 153 | 173 | 193 |
| 14 | 34 | 54 | 74 | 94 | 114 | 134 | 154 | 174 | 194 |
| 15 | 35 | 55 | 75 | 95 | 115 | 135 | 155 | 175 | 195 |
| 16 | 36 | 56 | 76 | 96 | 116 | 136 | 156 | 176 | 196 |
| 17 | 37 | 57 | 77 | 97 | 117 | 137 | 157 | 177 | 197 |
| 18 | 38 | 58 | 78 | 98 | 118 | 138 | 158 | 178 | 198 |
| 19 | 39 | 59 | 79 | 99 | 119 | 139 | 159 | 179 | 199 |
| 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 | 180 | 200 |

COMPUTER

**Reader Service Card**

**December 1990 issue** (Card void after June 1991)
Print address below.

Name ______________________

Title ______________________

Company ______________________

Address ______________________

City ______________________

State/ZIP ______________________

Country ______________ Phone ______________

**Please send** (Circle those you want)

**201** Publications catalog
**202** Membership information
**203** Student membership application
**204** Application for senior membership

**Reader interest**
(Circle what you liked; add comments on the back)

**A** Parallel Languages
**B** Compiling Scientific Code
**C** Parallel Computation
**D** Engineering Systems
**E** Benchmark Overview
**F** President's Message
**G** EIC Message
**H** Standards
**J** Update
**K** CS News
**L** Product Reviews
**M** New Products
**N** ICs/Microsystems
**P** Conferences
**Q** Call for Papers
**R** Calendar
**S** Book Reviews
**T** CS Magazines
**U** Annual Index

**Product information**
(Circle numbers for products and advertisements on which you want more information)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 21 | 41 | 61 | 81 | 101 | 121 | 141 | 161 | 181 |
| 2 | 22 | 42 | 62 | 82 | 102 | 122 | 142 | 162 | 182 |
| 3 | 23 | 43 | 63 | 83 | 103 | 123 | 143 | 163 | 183 |
| 4 | 24 | 44 | 64 | 84 | 104 | 124 | 144 | 164 | 184 |
| 5 | 25 | 45 | 65 | 85 | 105 | 125 | 145 | 165 | 185 |
| 6 | 26 | 46 | 66 | 86 | 106 | 126 | 146 | 166 | 186 |
| 7 | 27 | 47 | 67 | 87 | 107 | 127 | 147 | 167 | 187 |
| 8 | 28 | 48 | 68 | 88 | 108 | 128 | 148 | 168 | 188 |
| 9 | 29 | 49 | 69 | 89 | 109 | 129 | 149 | 169 | 189 |
| 10 | 30 | 50 | 70 | 90 | 110 | 130 | 150 | 170 | 190 |
| 11 | 31 | 51 | 71 | 91 | 111 | 131 | 151 | 171 | 191 |
| 12 | 32 | 52 | 72 | 92 | 112 | 132 | 152 | 172 | 192 |
| 13 | 33 | 53 | 73 | 93 | 113 | 133 | 153 | 173 | 193 |
| 14 | 34 | 54 | 74 | 94 | 114 | 134 | 154 | 174 | 194 |
| 15 | 35 | 55 | 75 | 95 | 115 | 135 | 155 | 175 | 195 |
| 16 | 36 | 56 | 76 | 96 | 116 | 136 | 156 | 176 | 196 |
| 17 | 37 | 57 | 77 | 97 | 117 | 137 | 157 | 177 | 197 |
| 18 | 38 | 58 | 78 | 98 | 118 | 138 | 158 | 178 | 198 |
| 19 | 39 | 59 | 79 | 99 | 119 | 139 | 159 | 179 | 199 |
| 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 | 180 | 200 |

COMPUTER

**Reader Service Card**

**December 1990 issue** (Card void after June 1991)
Print address below.

Name ______________________

Title ______________________

Company ______________________

Address ______________________

City ______________________

State/ZIP ______________________

Country ______________ Phone ______________

**Please send** (Circle those you want)

**201** Publications catalog
**202** Membership information
**203** Student membership application
**204** Application for senior membership

**Reader interest**
(Circle what you liked; add comments on the back)

**A** Parallel Languages
**B** Compiling Scientific Code
**C** Parallel Computation
**D** Engineering Systems
**E** Benchmark Overview
**F** President's Message
**G** EIC Message
**H** Standards
**J** Update
**K** CS News
**L** Product Reviews
**M** New Products
**N** ICs/Microsystems
**P** Conferences
**Q** Call for Papers
**R** Calendar
**S** Book Reviews
**T** CS Magazines
**U** Annual Index

**Product information**
(Circle numbers for products and advertisements on which you want more information)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 21 | 41 | 61 | 81 | 101 | 121 | 141 | 161 | 181 |
| 2 | 22 | 42 | 62 | 82 | 102 | 122 | 142 | 162 | 182 |
| 3 | 23 | 43 | 63 | 83 | 103 | 123 | 143 | 163 | 183 |
| 4 | 24 | 44 | 64 | 84 | 104 | 124 | 144 | 164 | 184 |
| 5 | 25 | 45 | 65 | 85 | 105 | 125 | 145 | 165 | 185 |
| 6 | 26 | 46 | 66 | 86 | 106 | 126 | 146 | 166 | 186 |
| 7 | 27 | 47 | 67 | 87 | 107 | 127 | 147 | 167 | 187 |
| 8 | 28 | 48 | 68 | 88 | 108 | 128 | 148 | 168 | 188 |
| 9 | 29 | 49 | 69 | 89 | 109 | 129 | 149 | 169 | 189 |
| 10 | 30 | 50 | 70 | 90 | 110 | 130 | 150 | 170 | 190 |
| 11 | 31 | 51 | 71 | 91 | 111 | 131 | 151 | 171 | 191 |
| 12 | 32 | 52 | 72 | 92 | 112 | 132 | 152 | 172 | 192 |
| 13 | 33 | 53 | 73 | 93 | 113 | 133 | 153 | 173 | 193 |
| 14 | 34 | 54 | 74 | 94 | 114 | 134 | 154 | 174 | 194 |
| 15 | 35 | 55 | 75 | 95 | 115 | 135 | 155 | 175 | 195 |
| 16 | 36 | 56 | 76 | 96 | 116 | 136 | 156 | 176 | 196 |
| 17 | 37 | 57 | 77 | 97 | 117 | 137 | 157 | 177 | 197 |
| 18 | 38 | 58 | 78 | 98 | 118 | 138 | 158 | 178 | 198 |
| 19 | 39 | 59 | 79 | 99 | 119 | 139 | 159 | 179 | 199 |
| 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 | 180 | 200 |

**Editorial comments**

*I liked:* ______________________________

______________________________

______________________________

______________________________

*I disliked:* ______________________________

______________________________

______________________________

*I would like to see:* ______________________________

______________________________

______________________________

______________________________

*For reader-service inquiries, see other side.*

PLACE
POSTAGE
HERE

***PO Box is for reader-service cards only.***

**COMPUTER**
Reader Service Inquiries
PO Box 16508
North Hollywood, CA 91615-6508
USA

**Editorial comments**

*I liked:* ______________________________

______________________________

______________________________

______________________________

*I disliked:* ______________________________

______________________________

______________________________

*I would like to see:* ______________________________

______________________________

______________________________

______________________________

*For reader-service inquiries, see other side.*

PLACE
POSTAGE
HERE

***PO Box is for reader-service cards only.***

**COMPUTER**
Reader Service Inquiries
PO Box 16508
North Hollywood, CA 91615-6508
USA

**Editorial comments**

*I liked:* ______________________________

______________________________

______________________________

______________________________

*I disliked:* ______________________________

______________________________

______________________________

*I would like to see:* ______________________________

______________________________

______________________________

______________________________

*For reader-service inquiries, see other side.*

PLACE
POSTAGE
HERE

***PO Box is for reader-service cards only.***

**COMPUTER**
Reader Service Inquiries
PO Box 16508
North Hollywood, CA 91615-6508
USA

# UPDATE

Contributions to Update are welcome. Send news of public policy or professional issues and of industrial or university research to Bob Carlson, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264, or to Jon T. Butler, Associate Technical Editor, Naval Postgraduate School, Code 62-Bu, Monterey, CA 93943-5004, e-mail butler@cs.ece.navy.mil

## Conflicts ensue over software "repossession," termination of service

*Bob Carlson, Staff Editor*

The rights of users and vendors came into sharp conflict recently in two cases that may force a reassessment of current practices in software licensing. In the first case, a California court is being asked to decide whether users of high-priced custom software who fall into arrears on their payments will have to live with the fear of a crippling phone call in the middle of the night. The second case involves the conditions under which a provider of electronic communications services is entitled to terminate service to subscribers.

**Lawsuit filed over software "repossession."** On the night of October 15, Logisticon, Inc., of Santa Clara, California, used telephone lines to disable inventory control software at two warehouses belonging to Revlon, the huge cosmetics firm, over a payment dispute.

Revlon had canceled the second phase of its agreement with Logisticon and was withholding payment on a $1.2-million contract, charging that the software did not work properly. Logisticon claimed that bugs in the system were minor and did not interfere with operation.

According to Logisticon President Don Gallagher, in a report by the *Wall Street Journal*, Revlon demanded free access to the source code in exchange for the $180,000 still owing on the contract. Logisticon employees dialed in and used computer access codes to disable the software when "we determined we had no recourse remaining," Gallagher said. He added that the system was rendered inoperable without harming Revlon's data.

Three days later, Logisticon turned the software on again. In the meantime, according to Revlon, daily sales activities totaling millions of dollars came to a standstill at the two distribution centers.

**Subscribers charge censorship.** Several subscribers to the Prodigy communications service accused the firm of selectively terminating their service after they used Prodigy's electronic mail service to enlist support for a revolt against increased charges by the company. Brian Ek, a spokesman for Prodigy, told the *Los Angeles Times* that the subscribers were terminated because the mass mailings are not allowed and amounted to harassment of other members. Yet at least one disconnected protester denied harassing anyone or sending mass messages.

Protest coordinator Russ Singer accepts Prodigy's elimination of obscenities or solicitations, according to the *Times*, but he charges that "they are editing the letters on the public bulletin board for content." He said that service to protesters was terminated after they revealed the extent of their support against higher fees to the service's on-line merchants and advertisers.

While declining to discuss specific cases of harassment, Ek cited high usage by subscribers and millions of dollars a year in operating costs to justify the increases. "This is not like a car where you buy it once and you own it," he told the *Times*. "It is a service."

**A collision of rights.** It appears that once again the legal profession is going to be called on to bring order to a relatively uncharted frontier that keeps expanding with the growing capabilities of technology. In reference to the Revlon incident, Robert J. Melford, who chairs a computing ethics subcommittee for the IEEE Computer Society, said it isn't uncommon for vendors to include a "time bomb" in their software scheduled to disable operation unless the vendor provides a key upon mutual agreement to extend the contract. Of course, all this should be stated up front so that it is clear to both parties.

The current case takes this concept a step further, however, and the outcome will depend in part on what was explicit in the contract between Logisticon and Revlon and whether the vendor violated its access privileges. "This area isn't well defined yet in law," added Melford, whose firm, Robert J. Melford Associates in Mission Viejo, California, consults in project management for systems analysis, design, integration, and security.

In the Prodigy case, Melford said, the issue hinges on who has the right to interpret the word "harassment." If the Prodigy service is dominant enough to be considered a monopoly, it might be classified as a utility rather than as a business. This would mean that ultimately an outside agency could be charged with regulating it for the public good.

Tim Headley, a patent law attorney with the firm of Baker and Botts, pointed out that a customer is subject to the loss of a software license if found to be violating its provisions. Pertinent questions in the Logisticon case are whether the firm had authorized permission to log in and what that permission covered. He added, however, that the law allows an authorized agent to do anything except "breach the peace" in executing a repossession.

**Get it in writing.** Melford sums up the current situation as a combination of separate issues that have been dealt with individually but never together. So while the ethics and legalities of "repossession" tactics and termination of service are being argued in court, and no doubt throughout the computer industry, customers are expected to start paying more attention to their contracts. Users of custom software, in particular, will want to specify in advance what dial-in access does and does not allow.

## Council formed to support software industry integrity

Citing the evolution of business practices detrimental to the entire industry, the newly formed Software Business Practices Council defined its goal as promoting ethical business practices and higher business standards.

"Neither the best people in the world nor the most spectacular technological achievements can ever overcome the damage to a reputation tarnished by dubious business practices," said Jeffrey P. Papows, chair of the new council and president and chief operating officer of Cognos, Inc.

Chief among the problems identified by Papows are announcements of "vaporware," insupportable and misrepresented marketing claims, and inconsistent reporting of software vendors' financial information. "Unless the software industry stops alienating and confusing customers and engendering cynicism, the reputation of the industry as a whole will deteriorate and growth will suffer," Papows said. "Clear lines of ethical behavior should be drawn by those of us in this industry, rather than by those who stand outside it and are less likely to appreciate its needs and dynamics."

The council offered several specific recommendations:

- Recognition by software vendors of distinctions between a product announcement and a statement of direction.
- Use of present and future standards in software product performance measurement.
- Support for efforts to achieve inexpensive user verifiability of vendor implementations of product performance measurement standards.
- Truthful, accurate, and verifiable explanation of product performance measurement standards in any comparative marketing and advertising claim.
- Public adherence to present and future standards of the Financial Accounting Standards Board of the Financial Accounting Foundation, and the American Institute of Certified Public Accountants (AICPA).
- Software industry vendor review of AICPA's proposed position statement on software revenue recognition. Companies are urged to submit comments about the content to both the AICPA and other regulating organizations.

The Software Business Practices Council is a nonprofit trade association. The founding member companies are AI Corp., Ashton-Tate, Banyan Systems, Chipcom, Cognos, Datamedia, Digital Equipment Corporation, Hewlett-Packard, Ingres, Intec Controls, Integral, Interleaf, Lotus Development, Multiview, Price Waterhouse, Ross Systems, and Sybase.

Members of the software vendor community are invited to participate by contacting Papows at Cognos, Inc., 67 South Bedford St., Burlington, MA 01803, phone (617) 229-6600, ext. 441.

# Computer Society NEWS

Editor: Guylaine M. Pollock, Sandia National Laboratories, Division 1424, PO Box 5800, Albuquerque, NM 87185; phone, (505) 846-0040; Internet, gmpollo@sandia.gov

## Society members elect new officers and board members

Results have been announced in the IEEE Computer Society's fall 1990 election. Voters chose Bruce D. Shriver to serve as president-elect for 1991 and president in 1992. He will succeed Duncan H. Lawrie, the society's 1991 president. The results, with the number of votes received shown after each name, were as follows:

| | |
|---|---|
| *President-elect:* | |
| Bruce D. Shriver (elected) | 6,553 |
| Joseph E. Urban | 4,209 |

**Vice presidents.** Paul L. Borrill was elected first vice president and Barry W. Johnson was elected second vice president. Both will serve one-year terms beginning January 1. Voting results were as follows:

| | |
|---|---|
| *First Vice President:* | |
| Paul L. Borrill (elected) | 6,852 |
| Gerald L. Engel | 3,681 |
| *Second Vice President:* | |
| Barry W. Johnson (elected) | 5,583 |
| Mario R. Barbacci | 4,764 |

**Board of Governors.** Seven of the 12 candidates for the Board of Governors were elected to serve three-year terms beginning January 1. The results and the number of votes received were as follows:

| | |
|---|---|
| *Elected to three-year terms (1991-93):* | |
| Anneliese von Mayrhauser | 6,303 |
| Fiorenza Albert-Howard | 6,196 |
| Benjamin W. Wah | 6,095 |
| Yale N. Patt | 5,878 |
| Ronald Waxman | 5,076 |
| Michael C. Mulder | 4,901 |
| Jon T. Butler | 4,674 |
| *Not elected:* | |
| Joseph Boykin | 4,648 |
| Donald E. Thomas | 4,582 |
| Akihiko Yamada | 4,572 |

**Bruce D. Shriver (left) was chosen as president-elect for 1991 and will serve as Computer Society president in 1992. Duncan H. Lawrie, current president-elect, starts his term as president January 1.**

**Table 1. Comparative statistics for recent Computer Society elections.**

| Election Year | 1985 | 1986 | 1987 | 1988 | 1989 | 1990 |
|---|---|---|---|---|---|---|
| Ballots mailed | 65,210 | 66,896 | 64,121 | 73,862 | 77,882 | 80,845 |
| Ballots returned | 9,314 | 10,080 | 9,047 | 10,110 | 10,263 | 11,174 |
| Percent responding | 14.3 | 15.1 | 14.1 | 13.7 | 13.2 | 13.8 |

| | |
|---|---|
| Michel Israel | 4,193 |
| Charles B. Silio | 3,974 |

**Constitutional amendments.** The suite of three constitutional amendments regarding presidential succession (*Computer*, Sept. 1990, p. 93), which required *for* votes by two thirds of the members voting, passed by a vote of 9,689 *for* and 681 *against*.

**Member participation in elections.** The rate of participation by eligible voters was up slightly for this year's election. Table 1 shows participation figures for the past six years.

**IEEE election.** As a result of the recent IEEE election, current Computer Society President Helen M. Wood will serve as Division VIII delegate-director on the IEEE Board of Directors for a two-year term beginning in 1991. Balloting results were as follows:

*Division VIII Delegate-Director:*

| | |
|---|---|
| Helen M. Wood | 6,156 |
| Bill D. Carroll | 3,755 |

## Board of directors dissolves AFIPS

The Board of Directors of the American Federation of Information Processing Societies voted at its October 13, 1990, meeting to dissolve the federation, effective immediately.

The IEEE Computer Society and the ACM, through their members on the AFIPS Board of Directors, took the lead in the organization's dissolution. In a joint statement, Computer Society President Helen M. Wood and ACM President John R. White affirmed the need for collective representation of US computing interests in the International Federation of Information Processing (IFIP). However, such representation "demands a federation very different from the AFIPS that has existed in the past . . . or even the AFIPS that would exist if its . . . structure and direction were to be retained," they stated.

AFIPS was created in 1961 as an umbrella organization of national computer societies representing the computing profession. It was best known for sponsoring the National Computer Conference (which ceased operation in 1987) and as the American representative to IFIP.

Wood emphasized that the move to dissolve AFIPS was a positive step in the history of joint professional computing activities in the United States. "We're simply abandoning an organizational form that served us well in the past but which required redesign for the 1990s and beyond," said Wood.

A computing federation of the future, as described by Wood and White, would not be involved in the dissemination and interchange of technical information, and its member organizations would not see it as a means for producing resources to support their own activities.

The Computer Society and ACM plan to sponsor a new organization that will enable US members of computing societies to take part in IFIP programs and activities. All eligible computing societies wishing to participate in these activities will be invited to join. Each society would pay only the costs directly associated with their participation, and the new organization will not be organized as a source of funds for any activities.

## Friends, associates mourn passing of Toy

Wing N. Toy, a member of the IEEE Computer Society Board of Governors and a long-time society volunteer, passed away Saturday, October 27.

In recent years, Toy served on *Computer*'s editorial board, as an associate technical editor for *IEEE Transactions on Computers*, and on the IEEE Ad Hoc Accreditation Visitors Committee. For his service to the society, he received the Meritorious Service Award in 1984 and the Certificate of Appreciation in 1987.

Toy, an IEEE fellow, was a supervisor in the Advanced Switching Networks Department at AT&T Bell Laboratories in Naperville, Illinois, where he was involved in the design of highly reliable processors for telecommunication applications for 36 years. He was named an AT&T Bell Labs fellow in 1983. He held 27 US patents and was coauthor of three textbooks on computer technology.

The family suggested that, in lieu of flowers, donations could be made to the American Cancer Society. The collection is being administered by Jill Leannah, AT&T Bell Laboratories, 200 Park Plaza, Indian Hill - IU203, Naperville, IL 60566.

**Wing N. Toy**

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC

INNS
INTERNATIONAL NEURAL NETWORK SOCIETY

The Official Airline
SINGAPORE AIRLINES

# IJCNN'91 SINGAPORE

INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS

WESTIN STAMFORD & WESTIN PLAZA - SINGAPORE, NOVEMBER 18-21, 1991

# *Call For Papers*

## CONFERENCE

The IEEE Neural Networks Council and the International Neural Networks Society (INNS) invite all persons interested in the field of Neural Networks to submit **FULL PAPERS** for possible presentation at the Conference.

**FULL PAPERS** must be received by 31 May 1991. All submissions will be acknowledged by mail. Authors should submit their work via Air Mail or Express Courier so as to ensure timely arrival. Papers will be reviewed by senior researchers in the field, and all papers accepted will be published in full in the Conference Proceedings. The Conference hosts tutorials on Nov 18 and tours arranged probably on Nov 17 and Nov 22, 1991. Conference sessions will be held from Nov 19-21, 1991. Proposals for tutorial speakers & topics should be submitted to Professor Toshio Fukuda (address below) by **Nov 15, 1990.**

## TOPICS OF INTEREST

Original, basic and applied papers in all areas of Neural Networks & their applications are being solicited. **FULL PAPERS** may be submitted for consideration as oral or poster presentations in (but not limited to) the following sessions:

- *Associative Memory*
- *Electrical Neurocomputer*
- *Image Processing*
- *Invertebrate Neural Networks*
- *Machine Vision*
- *Neurocognition*
- *Neuro-Dynamics*
- *Optical Neurocomputers*
- *Optimization*
- *Robotics*
- *Sensation & Perception*
- *Sensorimotor Control System*
- *Supervised Learning*
- *Unsupervised Learning*
- *Neuro-physiology*
- *Hybrid System (AI, Neural Networks, Fuzzy System)*
- *Mathematical Methods*
- *Applications*

## AUTHORS' SCHEDULE

| | |
|---|---|
| Deadline for submission of **FULL PAPERS** (Camera ready) | 31 May 1991 |
| Notification of acceptance | 31 Aug 1991 |

## SUBMISSION GUIDELINES

Eight copies (One original and seven copies) are required for submission. Do not fold or staple the original, camera-ready copy. Papers of no more than 6 pages, including figures, tables, and references, should be written in English and only complete papers will be considered. Papers must be submitted camera-ready on 8 1/2" x 11" white bond paper with 1" margins on all four sides. They should be prepared by typewriter or letter quality printer in one-column format, single spaced or similar type style of 10 points or larger and should be printed on one side of the paper only. FAX submissions are not acceptable. Centred at the top of the first page should be the complete title, author name(s), affiliation(s) and mailing address(es). This is followed by a blank space and then the abstract, up to 15 lines, followed by the text. In an accompanying letter, the following must be included:

| *Corresponding author:* | *Presentation preferred:* | *Technical Session:* | *Presenter:* |
|---|---|---|---|
| Name | Oral | 1st Choice | Name |
| Mailing Address | Poster | 2nd Choice | Mailing Address |
| Telephone & FAX number | | | Telephone & FAX number |

For submissions from Japan send to:
Professor Toshio Fukuda
Programme Chairman
IJCNN'91 SINGAPORE
Dept of Mechanical Engineering
Nagoya University, Furo-cho, Chikusa-Ku
Nagoya 464-01 Japan.

Fax: 81-52-781-9243

For submissions from USA send to:
Ms Nomi Feldman
Meeting Management
5565 Oberlin Drive, Suite 110
San Diego CA 92121.

Fax: 619-535-3880

For submissions from rest of the world send to:
Dr Teck-Seng, Low
IJCNN' 91 SINGAPORE
Communication Intl Associates Pte Ltd
44/46 Tanjong Pagar Road
Singapore 0208

Tel: (65) 226-2838
Fax: (65) 226-2877, (65) 221-8916

Editor: Richard Eckhouse, UMASS-Boston, Harbor Campus, Boston, MA 02125, Compmail II, r.eckhouse; Bitnet, eckhouse@umbsky; Compuserve, 70516, 556

# A holiday parade of products

Every year we fill the contents of our December column with a number of products we judge suitable for stuffing into your holiday stocking. We continue this tradition with a new set of items that we hope will surprise and delight you.

## How to label nearly everything

Lift Off is one of those products that is both simple and elegant, a product that any one of us could have made — if only we had the smarts to think of it. It's easy and fun to use, quickly enhances your presentations, and produces very professional results.

What is it? Lift Off is a professional lettering system that you can use on file folders, charts, technical drawings, proposals, blueprints, newsletters, notebooks, diplomas, and just about anything you can think of. It consists of software that produces horizontal or vertical lettering, special paper, transfer tape, and burnishing tools.

To run "3-2-1 Lift Off," you'll need an IBM PC or compatible, 320 Kbytes of memory, a graphics display card, a hard disk, and a laser printer that is PCL-compatible. The software is menu driven, with menu entries keyed to the PC's 10 function keys. A mouse is a useful but unnecessary option for selecting items from the menu.

The program always starts in edit mode. The main menu appears at the top of the screen, and a tape window in the middle of the menu provides a preview of how the Lift Off tape will look when it is printed. An edit window at the bottom of the screen allows you to enter and edit the tape text.

Selecting a menu item by pressing a key or highlighting it with the mouse causes another menu to appear that offers further choices. The PC's standard cursor keys let you move within the edit window and change from insert to replace mode.

The main menu items allow you to set type attributes. These specifications vary from serif or block font in normal or bold (with a horizontal or vertical orientation) to a point size from 6 to 60. You can also load and save previous text, set up and use the laser printer, place a border around text that can be underlined and printed as a normal or an inverse image, adjust the space between characters (both manually and automatically), clear the text window, and exit to DOS. Maximum length for a tape is limited to letter-size paper, but within that working limit you can produce any combination of letters and symbols in various type faces, sizes, degrees of boldness, and orientations.

The Lift Off paper is specially coated on both sides; the laser toner does not adhere to it. A tape message prints on the left side of the page, allowing you to turn the paper around for a second printing, or over for two more passes. Since each of the four printing areas can be used from three to six times, an estimated yield of 12 to 24 copies per sheet is possible.

Having printed a message on the page, it is a simple matter to select one of the transparent tapes (essentially cellophane tape in two widths and two types: clear or frosted) to apply over the message and accept the toner. A burnishing tool and pad are included within the kit. A nice touch is that the burnishing pad includes samples that show off most of the features I've mentioned. The transfer tape does not stick to the Lift Off paper so—after burnishing the toner into the tape—you peel it off and apply it to the object to be lettered.

All of this reminds me of the Kroy lettering system, but what makes this one so much more clever is that the toner is applied to the sticky or bottom side of the transfer tape, rather than the top as in the other system. The results are thus much less likely to be scratched off, even with heavy use.

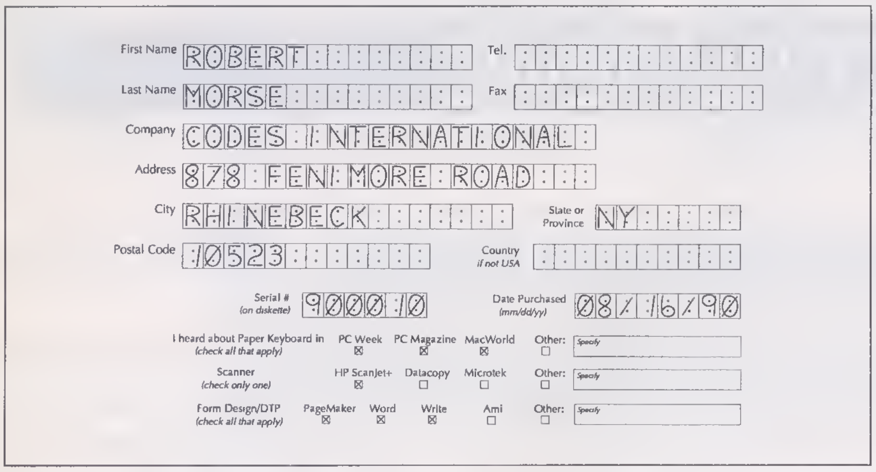
While the basic system comes with serif and block lettering, as well as seven special symbols, a Lift Off font pack includes three additional fonts and a collection of 93 special symbols. At $49, the font pack is worth the extra cost to the base price of $89 for the Lift Off kit. For another $49, you can purchase a supplies pack that includes 50 additional sheets of Lift Off paper, another burnishing tool, and six more rolls of Lift Off tape. (You may not need those 50 sheets immediately because you are treated to an additional 12 sheets as an inducement to returning your registration card.)

The instruction guide that comes with the product is well done and includes many screen shots, detailed instructions, and an index. I have no doubts that this is a useful product. I already have found dozens of ways to use it. Contact DP-Tek, Inc., 3031 W. Pawnee, Wichita, KS 67213, phone (800) 727-3130, to order a copy. — *R. Eckhouse*

**Reader Service 21**

## A computer that makes and reads questionnaires

Handwritten character input to computer applications has been the illusive goal of those trying to offer forms that are easy to fill in. This is not to mention the hurdles associated with producing accurate recognition and compatibility with other software applications. Solutions have been relatively expensive up to now, requiring special hardware and software that often runs on machines considerably bigger than PCs. Datacap is in the process of changing that with



First Name ROBERT
Tel.
Last Name MORSE
Fax
Company CODES INTERNATIONAL
Address 878 FENIMORE ROAD
City RHINEBECK
State or Province NY
Postal Code 10523
Country *if not USA*
Serial # *(on diskette)* 9000 10
Date Purchased *(mm/dd/yy)* 08/16/90

I heard about Paper Keyboard in *(check all that apply)* PC Week ☒ PC Magazine ☒ MacWorld ☒ Other: ☐ *Specify*
Scanner *(check only one)* HP ScanJet+ ☒ Datacopy ☐ Microtek ☐ Other: ☐ *Specify*
Form Design/DTP *(check all that apply)* PageMaker ☒ Word ☒ Write ☒ Ami ☐ Other: ☐ *Specify*

**A sample form for use with Paper Keyboard.**

Paper Keyboard, a package that makes it easy to design forms, fill them in, scan and validate the results automatically, and export the data to popular databases.

Paper Keyboard is an interesting blend of almost all the right ingredients in a package that runs on either the Macintosh or an IBM PC/compatible. In the case of the PC version, reviewed here, Paper Keyboard harmoniously marries Windows 3.0 with the features of a GUI word processor (I used Ami Pro) and the HP Scan Jet — in one $895 package. It is so easy to use that I went from reading the manual to producing useful results in just a couple of hours.

Typically, when you set up a computer-readable form, you start by thinking about entry fields and check boxes. The entry fields are for names, addresses, dates, phone numbers, and questions that require more than a yes or no response. Here Paper Keyboard recognizes the alphanumeric characters along with five punctuation characters including the comma, period, slash, dash, and apostrophe. But when all you need is a yes-or-no, one-from-many, or even many-from-many (such as marking all that apply) response, a check box is offered as an alternative. And when your response is "other" and you want to write in an unanticipated response, Paper Keyboard offers you operator-filled-in fields; that is, fields that will be keyed in by the person processing the form.

Once you have developed a form, the next step is to use a word processor or desktop publisher to actually produce it. While it would be nice if you could produce a formless form, reality requires that the form have specific fields that are fixed rigidly on the page. In the case of Paper Keyboard, data is entered into rectangular boxes called dominos because they contain two dots that help the user align the character to be written. A string of domino characters makes up a field and usually has associated text nearby to indicate what is to be filled in, such as a last name or zip code.

The two dots or eyes within the domino serve as the discriminators for correctly recognizing a character. Still, there can be particular difficulties with the letter O and the numeral 0; U and V; S and 5; M, N, and W; and A and R. In fact, the number I and the letter I are only distinguished in an alphanumeric field by their position relative to the two eyes (the I is to the left of both eyes, and the 1 appears to the right).

Paper Keyboard supports this process of making up a form by including six fonts in sizes of 16, 20, 24, 30, and 36 points. They are installed in Windows through the control panel and are used both for on-screen display and for printed laser output of the actual form. These sizes are appropriate for cramming a lot of fields onto the form and allowing easy user entry into the fields. The font set includes both dominos and check boxes, and each can be empty or filled. Empty fields serve for data input, while filled fields offer guidance in how characters are to be written in. Finally, anchors or cross hairs are placed on the form so the system can register the page and correct for tilted or off-center scans.

After producing the form, the next step is to create a form specification, or the Form Spec, which tells Paper Keyboard how to scan the form by locating fields and check boxes. It also helps validate the entry information, such as numerics in an alpha field or more than one box checked when only one is allowed. The Form Spec is like a C program and provides details about the form size, field locations, data exporting, and conditionals for field checking.

As with everything else about Paper Keyboard, you realize that the designers made every attempt to offer flexibility with ease of use. In the case of the Form Spec, however, they really let us down with regard to specifying where fields are located on the form. You literally have to take out a ruler and measure each form and field in the form in tenths of a millimeter to determine how big the form is, where the anchors are located, and where each field coordinate begins and ends. Clearly, the graphical interface of Windows on which they built this product should have made it a snap to generate this portion of the Form Spec. Since Paper Keyboard displays where it thinks these fields are by superimposing Form Spec locations on the form, it is surprising that the contents of the Form Spec cannot be changed by using this information.

Once a form is filled in, the next step is to scan it in and recognize the many fields, checking the boxes that comprise a form. This is where Paper Keyboard comes into play. Using a familiar Windows display, Paper Keyboard offers four icons on the left of the screen and five menu items across the top. The icons are used to open the Form Spec, scan a form, recognize the contents of the form, and zoom in or magnify the scanned image. The first three steps can be automated with a sheet-feeding scanner.

Two conditions cause Paper Keyboard to pause: when the program tries to recognize a character and comes up with a confidence level below some preset value, or when an operator field must be keyed in from a handwritten, user-supplied field. Editing and accepting each character or field is easy, greatly aided by the zoom capability; the Form Spec selects what is to be displayed when Paper Keyboard pauses for human intervention.

After accepting the fields within a form, the operator exports the data for use in some external program or database. Again, the Form Spec is used to delineate how the data is exported. At the present time, Paper Keyboard does not support any particular database or spreadsheet, and instead puts out the results as pure ASCII strings. Depending on how you look at this, it can be a blessing or a pain.

Paper Keyboard is an exciting product. It's an intelligent way to use existing resources to solve the problem of handwritten input to a computer. The recognition software works well as long as those who fill in the form remember to print the characters in the style that the software requires. In terms of thorough system documentation and completeness of the product, the folks at Datacap have done their homework. I think that I will hold out until they automate the Form Spec process and offer an integration path into some of the more popular database systems. From what I can gather, both are in the works and may well be available by the time this review appears.

You may reach Datacap Inc. at 5 West Main St., Elmsford, NY 10523, phone (914) 347-7133. — *R. Eckhouse*

**Reader Service 22**

## A perfect 10

Measured in terms of versatility, version 1.2 of Super Base 4 rates a 10. Not only does Super Base 4 provide in Windows 3.0 all the features of a mainframe relational database management system, but it does so while maintaining the simplicity of a PC flat-file database system. It also supports the Windows 3.0 DDE protocol, references graphics and text files, and imports data from a variety of spreadsheets — as well as dBase II and III databases. The program offers a security system to control who can read, write, or delete data. And, finally, it comes with a full-featured forms editor.

A Super Base 4 database is a mostly flat file. Each database has a fixed number of fields of three different types: character strings, numbers, and date times. Character strings can range from one to 4,000 characters. There are a number of filters that can be set to make sure that the data has a consistent format. These filters can convert the input data to a variety of formats (including all upper case and all lower case), capitalize the first word in the field, or capitalize every word in the field. Strings are stored in the database at their actual length so there is no wasted space when a short string is stored in a long field.

Numbers, except money, are stored with 14 digits of accuracy. However, they can be displayed with any number of digits to the right and left of the decimal point and with a wide range of formats such as leading or trailing zeros, sign to the right or left of the number, negative numbers in parentheses, or even a comma for use in numbers that have more than three digits.

Numbers with a display format that specifies two digits to the right of the decimal point are considered money. Money is stored with this format to prevent rounding errors. Each number can be displayed with a currency sign or a percent sign. The currency sign used in the database can be changed to any character that you want. Dates can be displayed with a space, slash, comma, hyphen, or period as the separator. The order of the year, month, and day can change. The year can be represented as two or four characters. The month can be a number, three character abbreviations, or fully spelled out. A time can be associated with the date, can include milliseconds, and can be displayed in a 12- or 24-hour format.

I described the database as "mostly" flat because there are a few bumps. A character string can be treated as the name of a file. This file displays in a separate window and can be either a text or graphics file. The graphic formats supported are IMG, PCX, TIFF, and WMF. Text files can be searched for key words just like text fields. Another bump is that character fields may be repeated. Multiple instances of the same field in a single record can occur, for example, in a dependent's field in an employee record. The only limitation is that the total length of all instances cannot exceed 4,000 characters.

The product also contains extensive tools for linking several databases. A database can display data from another database and use it to validate user input or in calculations for virtual fields. Virtual fields are calculated when the record is displayed; they are not stored in the database.

Database creation is straightforward. A series of dialogue boxes guides you from the database path entrance (by means of password selection) to field definition to index selection. You can also modify the database structure at any time. You can add fields, delete them, or change their data type. When a data type changes, any existing data is converted. When the data in a field cannot be converted to the new type, the field's value is set to null.

Super Base 4 has an easy-to-use and consistent user interface. Its main claim to fame is the row of buttons at the bottom on the window. These buttons resemble the controls on a VCR. Super Base 4 treats each record like a frame in a VCR tape, and you can play the "tape" either backward or forward — sort of like the VCR search mode. The time that each record displays can be adjusted to match the record length and your reading speed. Some buttons (like the filter and redisplay buttons) do not have any VCR equivalents. I found that the interface was not as intuitive as advertised, but was nevertheless easy to learn and use.

There are three built-in display modes: table, page, and record. A forms mode displays forms that you build with the Super Base 4 editor. The table mode displays one record per line; it's the traditional flat-file view. Record mode displays one record at a time in the window. Each field has its own line in the window. Page mode starts out like record mode, but you can point to a field and move it around on the page. The new setup is recorded for use the next time the database is displayed in page mode. It's a very simple forms editor. Forms mode displays a form built with the editor. It allows you to draw lines and boxes around groups of fields, position titles, and fields anywhere on the page and control the color and fonts with which the titles and fields are displayed.

For people who would rather write programs, Super Base 4 contains a very powerful data-manipulation language. This DML can be used to create or modify a database structure or insert, search, and update data. It includes an if-then-else structure, gosub, and for and while statements. You can also define local variables. There are even statements for querying the user via a dialogue box and displaying messages or records so that the programs are interactive.

I found the technical support for Super Base 4 to be very good. However, you only receive 30 days of free support, and the clock starts when they receive your registration card. I didn't need much technical support. Between the well-written manuals, the 11-lesson tutorial, and a very good hypertext-like help system, I could figure most things out.

The only real problem I found was importing dBase III databases with memo fields. There is no way to import the memo text into the Super Base 4 database. Instead, you get the memo index number. If it weren't for my large investment in dBase III databases with memo fields, I would be using Super Base 4 now. As it is, I plan to use it for any new applications.

Super Base 4 is available from Precision Inc., 8404 Sterling St. A, Irving, TX 75063, phone (214) 929-4888, at a list price of $695.— *N. Davids*

**Reader Service 23**

## Artisoft's Fantastic Lantastic

*Computer* was one of the first magazines to review Artisoft's Lantastic network operating system (NOS) and the 2-Mbps adapter card (see the August 1988 issue, pp. 90-91). Reviews have followed concerning the Network Eye and NOS updates (December 1989, pp. 81-90). Each review rated Artisoft as fantastic and praised it as an outstanding value that results from offering features found in much more expensive products at a price that anyone could afford. Equally important, my reviews have marvelled that NOS can run as a server with a small amount of memory (less than 40 Kbytes) or as a workstation with a miniscule 12 Kbytes. Taking less room than many popular TSRs, NOS runs transparently as a nondedicated server in a typical applications-based environment. Thus, a separate and expensive machine does not have to be dedicated to providing all system resources, and every machine can function equally well as a computing engine and a server.

Other independent reviews have also recognized Artisoft and heaped praise on the company, rating it a best buy and selecting it as an editor's choice. And, all the while, Artisoft has continued to develop and release a host of new hardware and software. In this review, I cover version 3.0 of the Lantastic NOS, the Lantastic voice adapter, and the high-performance 8/16-bit Ethernet adapter called the AE-2. Each one continues to rate as an outstanding value.

### Lantastic NOS, V. 3.0

Even from the beginning, Lantastic NOS was packed with a variety of features not found in other "low-end" packages. I particularly liked that NOS could run in the background and also run all of my standard applications such as word processing, spreadsheet, graphics, and schematic capture programs. I could operate NOS from the extensive command options that could be invoked in three ways: (a) from the command line, (b) from the set of menus that came with the Net and Net_Mgr programs, or (c) as a pop-up utility called LANPUP. Equally impressive was the full set of security controls along with audit trails for controlling network access. Of course, none of that has changed, but so much has been added in version 3.0 that this release is well worth the $50 upgrade fee for licensed users.

What are the changes? First, in this latest version, automatic installation generates the necessary changes to your CONFIG.SYS file as well as builds a startup batch file that both gets the network going and logs you into other nodes. The process is accomplished by running the install program and answering a few questions regarding what you want done and what nodes you want to automatically log into when you start up NOS.

Second, the NOS manual has been completely revised and split into two sections that are bound together. The user's manual is short (approximately 50 pages) and covers just what you need to get started. The reference manual is more than 200 pages and covers everything in greater detail. It is organized so that separate sections cover each of the functions and features to be found in NOS. The appendices cover such topics as improving network performance, setting up batch files to start up the nodes in the network (workstations or servers), and testing the network adapters. They also cover trouble shooting and list all the various system and error messages. Both manuals contain a table of contents and an index.

Third, there are a number of performance enhancements. While this seemingly brief description hardly does version 3.0 justice given the significance of these enhancements, I think you will get the idea. These improvements include

- despooling to more than one printer simultaneously (a feature generally reserved for the larger systems if found at all);
- moving the printer spool area to a faster disk, including a RAM disk;
- increasing the size of the network print buffers to speed up printing;
- caching software to speed up disk accesses with parameters to set the size, location, write delay, etc.;
- clearing the printer spool area with one command rather than deleting each file individually;
- redirecting serial printers at the server rather than using the mode command; and
- bypassing the DOS restriction of 255 open files by allowing you to open up to 5,100 files per server.

Fourth, network security has been enhanced by

- limiting user access to certain hours on certain days,
- setting expiration dates for both passwords and accounts,
- allowing backup and restoration of your control directory (where information about user accounts and server resources is kept),
- using indirect files that point to a file in another directory, and
- providing password protection for the Net_Mgr program along with context-sensitive help using the F1 key (a feature also found in the Net program as well).

Lantastic NOS also allows you to create multiple control directories so that you can have completely different user accounts and resources.

Fifth, additional functions and features are

- a pop-up email notification as well as a single-line message (like "phone" or "talk" utilities found in other networks),
- remote booting with the new AE-2 Ethernet card or the enhanced version of the 2-Mbyte/s boards (so diskless workstations do not even require a floppy to boot up), and
- improvements to LANPUP, the TSR utility that offers the features of the Net program (as a single line rather than as a menu-driven program).

By the way, LANPUP can be also run as a stand-alone program if you cannot spare the 5 Kbytes it takes as a TSR.

I should point out two caveats. One is that you will need DOS 3.1 or higher to run NOS. The other is that NOS no longer operates as a server under Windows 3.0 in enhanced mode. In the latter case, it worked just fine under Windows 2.1, but the latest version does not seem to be compatible with this fine network operating system. I'm not sure whose problem it is, but I do hope that Microsoft and Artisoft find a solution.

### Lantastic Voice Adapter

This small card operates in 8-bit mode. It digitizes voice using a sample rate of approximately 8 kHz. Compression cuts storage requirements in half without seriously compromising recognition. The board uses DMA channels 1 and 3 to provide full-duplex operation (such as simultaneous record and play). Either or both channels can be disabled, but you cannot run Voice Chat if the DMA channels are disabled.

When the board and Lanvoice software are installed, the Chat utility allows the user to record and play back voicemail messages. You can be notified of incoming mail via a pop-up note, and you can use the Net menu system to play or record messages. A special screen for voice mail operates in record, play, or pause modes and shows the message size in both bytes and minutes/seconds. Playback operates almost like a cassette recorder in that it can fast forward and rewind in steps of small, large, or 2-second increments

There is a separate Lantastic Voice Programmers Interface (VPI) that includes the commands necessary to provide direct support in applications. The eight basic commands are

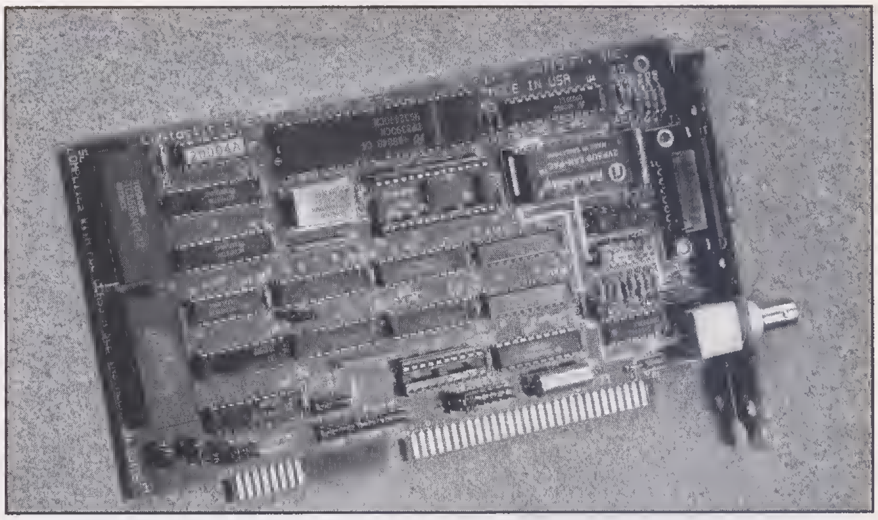
- status (to find out about the voice adapter),
- reset (to initialize the adapter),
- deinstall (to remove the software from memory),
- cancel (to remove a pending voice control block),
- send (to transmit a play channel buffer),
- silence (to send silence to the play channel),
- receive (to accept a record channel), and
- threshold (to wait for a sound to reach a certain loudness before beginning to receive from a record channel).

The VPI is sold separately for $195.

Record and Say utilities included with the adapter allow you to record and play back voice messages, which can then be embedded in other programs. For a test, I used Say within a Quick Basic program to prompt for input and then acknowledge if the required input was within a specified range. Another test was to make it part of the AUTOEXEC.BAT file so that the machine verbally prompted me when it was through with the boot procedure. Both worked easily and produced excellent results.

The $149 price for the product includes the board, a telephone handset and coiled cord, and the network software and utilities. In addition to the external connector for the telephone handset, two phono jacks for line in/out lines allow connection to an external microphone/amplifier.

This handy, versatile board offers voice for both network and nonnetwork applications. Operation is extremely simple and fully documented in the small user's manual that comes with the board.


**Lantastic Ethernet Adapter AE-2 board.**

## AE-2 Ethernet board

The latest network board for Artisoft is the AE-2 Ethernet card that runs at a full 10 Mbps and complies with the 802.3 standard. The board automatically switches from 16-bit to 8-bit modes when placed in an 8-bit slot, or you can force it into 8-bit mode by using one of the jumpers on the board. Other jumpers are set to specify the IRQ line (2-7, 10, or 15), the DMA channel (1, 3, 5, and 7, but currently not supported), the I/O port address for the 32-byte I/O space used by the adapter (300h, 320h, 340h, or 360h), and the Ethernet type (Cheapernet using an RG-58 coaxial cable or Ethernet using Ethernet transceivers). The jumpers also specify the boot ROM for diskless workstations, the Cheapernet segment length (165 or 300 meters with a limit of a single segment if 300 meters long), nonstandard bus selection for machines with incompatible bus timings, and NE2000 emulation to allow the use of Novel software such as TCP/IP or Netware.

If you want to use coax and repeaters, the maximum cable length is 185 meters with 30 adapters per segment. You can extend it to 300 meters if you give up the need for repeaters and hence 802.3 compliance. If you switch to 10Base5 mode in which you use an Ethernet transceiver, you can achieve a 500-meter cable length and up to 100 nodes per segment.

The board comes with 16 Kbytes of RAM that can be expanded to 64 Kbytes as an option. A second option priced at $99 includes the boot ROM for diskless workstations (unfortunately, I did not get to test this). A $725 starter kit includes two AE-2 boards, Lantastic's NOS, AI-LAN BIOS (the adapter independent version of Artisoft's LAN BIOS), 25 feet of coaxial cable, terminators, and full documentation. Additional boards are priced at $349. A Micro Channel version will be available after the first of the year.

In testing the system, I plugged the AE-2 cards into the same machines with the older 8-bit Ethernet adapters installed, cabled up the systems, and ran the same Lantastic NOS software I had been using. The only noticeable change was that everything ran at least twice as fast! I have to admit I didn't run any performance tests, but it was clear that the 16-bit data path to the board and the higher data-transfer rates do make for noticeably speedier transfers.

Like all the boards I have reviewed from Artisoft, the AE-2s are well made and compact. About the only thing one could suggest to Artisoft is to combine the Voice Adapter with the AE-2 on the same card so that you don't have to give up two slots to install them both.

## Summary

My personal computing environment has grown to include four machines: an 8086, a 286, a 386, and a 386 laptop (and a 486 will be added shortly). Without question, I find that a LAN is necessary for the transfer of data between machines. In fact, it makes economic sense because I would have to add disks and printers to each new machine if I continued to operate in a stand-alone fashion. In addition, the floppy disk situation would require multiple drives on each system. Also, since I use tape to back up my hard disks, I'd have to throw in those costs as well.

With Lantastic, I gain in several ways. First, there is the expense mentioned above. Second, there is the question of slots and space. Either I don't have any slots left in a machine, or I don't have room next to the computer for a printer. With Lantastic, I can tailor each node to the space within the machine and around it. Finally, there is the convenience of sharing applications and data between machines. This means a lot less duplication and a lot more free disk space. Thus, even if you have needs for a small LAN, such as I have, the price/performance of a Lantastic system makes good sense; I don't think you will find anything like it anywhere else.

But there are other reasons to go with Artisoft. Obviously the company is continually developing newer and better products. These products generally replace older ones with better performance at the same price. The company also seems tuned in to customer needs. For example, it now offers free, unlimited technical support to registered end users, doing away with the older plan that required you to get support from the selling dealer or pay $100 per year for direct support. And most important, Artisoft offers full-featured products at prices less than you would expect to pay; you get the functionality of the higher priced systems at rock-bottom prices.

As you can tell, Artisoft is tops on my list of suppliers of LAN equipment. I have recommended the company without reservation. Feedback from users who took my advice has been overwhelmingly enthusiastic. Thus, I, too, add to the chorus of users who rate Artisoft boards a best buy and the number one choice when it comes to LAN systems. In fact, in expanding my system, I bought the additional adapters. What better recommendation than that can I give?

Readers may contact Artisoft, Inc. at Artisoft Plaza, 575 E. River Rd., Tucson, AZ 85704, phone (602) 293-6363, fax (602) 293-8065. — *R. Eckhouse*

NOS **Reader Service 24**
Voice Adapter **Reader Service 25**
Board **Reader Service 26**

## Inside Windows 3.0

When running Windows 3.0, have you ever wondered if your display board requires scaling support? Or how much linear space is under DPMI management? Or the size of the code or data blocks of the currently running tasks? No? Well, I have to admit I haven't either, but all this information and much more is available about using Win Sleuth from Dariana Technology Group Inc., 6945 Hermosa Circle, Buena Park, CA 90620, phone (714) 994-7400, fax (714) 994-7401.

Win Sleuth is the latest diagnostic software package from a family of such products that includes a similar package for PC/MS-DOS and even the Macintosh. Version 1.0 reviewed here includes 11 modules that describe general system information, power-on.In status, hard disk characteristics, display attributes, MS-DOS information, memory allocation sizes, printer characteristics, allocation of system memory (between 640 Kbytes and 1 Mbyte), information on Windows, network characteristics, and finally suggestions for improving Windows' performance.

The amount of information provided depends on the sophistication level you set (novice, intermediate, or advanced). The colorful Windows display makes it easy to select which or all of the modules you wish to run, along with a choice of where to present the information (on the screen or to the default printing device).

In some cases, the information provided is generally known by the user, or is readily available. In other cases, it is information you might need to know but can't figure out how to find. So, while most users will know they have a 286 or a 386 and what floppy drives are installed, they may not know if a math coprocessor is present or what the base addresses are for their serial and parallel ports. And while they may not care about how many Windows tasks are running, knowing how real mode segments from C000-FFFF are used can be awfully important if you have a network or I/O card filling the space — and causing Windows to crash often. Things like knowing the DPMI memory statistics may not be important now, but with more and more new programs trying to cash in on Windows memory management, it can suddenly become very important when things go haywire.

Win Sleuth is easy to install and use. It includes a brief — but complete — user's manual and on-line, context-sensitive help. You won't use this program often, but its value will be evident when it saves you from opening up your box to find out just what is under the hood. This is particularly true when you add new options, both software and hardware, to your machine. My only complaint about Win Sleuth is that it retails for $149, which I find a bit expensive in comparison to other packages. A very nice feature is that it runs in any mode that Windows runs in: real, standard, or enhanced. — *R. Eckhouse*

**Reader Service 27**

## State of the art in forms packages

Most of us are not form designers, but we are form "fillers." Expense reports, tax payments, insurance claims —what have you. We fill them in all the time. While filling out these forms, we often wonder whether we could design better ones. With Per Form from Delrina Technology, you can design the form and automate filling it in, thus simplifying the process, automating record keeping, and generating very professional results.

Of all the products I have reviewed, I have spent the most time on Per Form. Not because it's complex or difficult to master. Quite the contrary. Per Form is easy and fun to use, well documented, complete, and very provocative. Therein lies the problem: I can't seem to tear myself away from this software.

I started out looking at an earlier version of Per Form that ran under the GEM/3 environment. I immediately liked what I saw but held off my review until the latest version, Per Form Pro, became available. This version runs under Windows 3.0, my favorite windowing environment, with the result that what was good before seems spectacular now. You have full Windows support plus a little bonus: You get a copy of Agfa Compugraphics Type Director with two typefaces (and hence multiple fonts). In addition, printing speed has been greatly improved, and sophisticated security features have been added. Also new are form folders and multipage forms. With the inclusion of color, object grouping, and a sophisticated set of calculation functions, designing forms is quite easy. And, filling out the form as well as linking it to dBase and ASCII file formats is easier. It is, put simply, a flawless implementation of a high-end system that appears to be designed by users for users.

But here I go telling why I like it even before I tell you what it is. For those of you unfamiliar with a form designer, let me say it is a combination of a graphics program and a database application. In the forms designer, you have both the menu bar and the tool box. The tools include the pointer to select menus, commands, options, objects, and areas. The I-beam tool selects where you want to insert or highlight text. The Line tool comes in two forms: one to draw lines at any angle and the other to restrict them to the horizontal or the vertical axis. You create boxes with the Box tool, which also comes in two forms, one with square and the other with rounded corners. Likewise, the Text area tools are for text inserted as a part of the form or for text to be filled in using the forms filler. A Graphic tool creates the frame into which you can import TIFF, IMG, PNT, PCX, GEM, BMP, WMF, and EPS images. Interestingly enough, you can import a graph in traceable form, allowing you to recreate it by drawing on top of it.

These tools should already be familiar to users of most paint programs. What's unusual and necessary for the form designer are the Comb and bar code tools. The Comb tool creates rectangular objects with multiple, evenly spaced partitions (horizontal or vertical) that can be patterned (like green bars on printer paper) and enclosed within a border. The bar code tool creates bar codes (in eight popular formats) that are bound during the design of the form or are created when the form is filled in.

The menu comes into play during the design process. The File menu includes the usual stuff common to Windows programs plus the capability to lock down a form (that is, save it in a form that can't be changed) and merge one form with another form. When you save a form you can also use the form information button to fill in the name of the designer, title, version, last revision date, and a form description. I'm beginning to see this feature in a number of Windows products, and I hope that others will follow suit because it alleviates the dependence on the archaic eight-character name and three-character extension for a DOS file.

The Edit menu is pretty typical but also lets you select all objects within the form. The Object menu allows you to display the attributes of, duplicate, position, align, repeat, lock, move to the front/back, make nonprintable, and border an object. There are secondary dialogue boxes for most of these commands, and the content of each depends on the object selected. The alignment command is set up to aid in aligning objects by top, bottom, left, right, and vertical/horizontal center. When you select an object, its type is shown on a separate line below the menu line along with the page number for the form (useful for multipage forms) and an asterisk indicator that lets you know when the form has been modified but not saved.

The View menu essentially lets you choose what is displayed on the screen, set preferences, and zoom in or out of your form. You use the Text menu to set how the text looks, is aligned, and is oriented. This menu also allows you to load text that was created externally. A thoughtful touch here is that the font dialogue box, as well as the position dialogue box from the Object menu, are "sticky." This means they remain on the screen until you close them. Thus, you can change and set fonts or positions, with results immediately available, even when working with a different menu. Now that shows off the power of a graphical interface!

**Of all the products I have reviewed, I have spent the most time on this one.**

The Fill menu specifies how the user is to fill in the form. A large number of decisions must be made here, from selecting field formats to an extensive set of calculations (like mathematical, logical, string, and financial operations) to the order for fields to be filled in. The Line menu includes standard or custom line widths that are either solid or patterned. The Shade menu determines the color of lines, backgrounds, and foregrounds, as well as fillable text, graphics, and bar codes. Notice that when a form is filled in, a unique graphic or bar code is generated from the user-supplied text.

The form filler is a separate program, callable from the file menu of the form designer (and vice versa). Menu choices include File, Data, Edit, View, Locate, Security, and Info. The file, edit, and view commands are similar to those found in the form designer. Data commands save, retrieve, and manipulate data files. The Locate commands are used to move through data records. Searching can occur on strings or by indices. Security commands control the access to specified forms and fields, as set up by the form designer. The Information commands provide customized form help, generalized program help, and lists of valid entries for fields. The combination of the Data, Locate, and Information commands makes the filler pretty powerful in terms of adding, modifying, and retrieving dBase files. When combined with linked fields within a folder, the result is akin to having an underlying relational database.

The manual set for this product is equally impressive. One manual, called Getting Started, explains Per Form and describes the installation process (which is automatic). The manual also goes through running Per Form and printing results, working with fonts, and networking. Appendices cover common problems, printing tips, and using Type Director.

The Form Designer manual is equally as thorough and covers the concepts of form design and using form designer, and includes a four-lesson tutorial. This manual also serves as a reference manual because it explains everything from the toolbox to the menu commands. Per Form supplies a large number (around 100) of sample forms that are useful for exploring the tutorial. These forms also serve as templates for you to make your own forms.

Finally, a smaller Form Filler manual offers chapters that contain a fast track for beginners and a list of commands, plus how to use the form filler. Tutorial lessons can also be found in this manual.

All manuals are indexed and well laid out, so it is easy to find specific help when you need it.

I used Per Form to design and print out an expense sheet for my company and for the two conferences I serve as treasurer. Like all good users, I avoided reading the manuals at first, only referring to them when I got stuck. I found the extensive on-line help system equally useful. In no time at all, I had some pretty fancy results printed out on my Kyocera laser and a burning desire to create more such forms. Whatever I wanted to do could easily be done, following the law of least astonishment: Things happen the way you think they should. Every time I thought of something that had been left out, a quick trip to the manual showed me that the designers had thought about it so that it had been built into the product.

Even after pretty extensive testing, a great many features remained that I never got around to using. One example is the folder feature that lets you store multiple forms within a folder so you can create an application with all the forms linked to the same database. Another is the security feature that locks forms and even fields within a form so one set of users can enter data while another can approve the information that is already filled in. Also, I did not test Per Form within a network environment. In fact, there is so much here that I doubt I will ever get to use all the features and functionality offered.

As far as I'm concerned, Per Form Pro is leading edge when it comes to both form design and form completion. It's a powerful, sophisticated package that is a steal at $495. An interesting accessory is the US Government Forms package, which includes 60 of the most commonly used DoD forms that Delrina claims are "precisely laid out to government specifications." But even if you don't need to design or fill in forms, you should look at this product as one of the best examples I've reviewed on how to create software. It is well designed and thoughtfully implemented, includes no surprises, makes efficient use of the user's time, and produces very professional results.

You can reach Delrina Technology Inc. at 15495 Los Gatos Blvd, Unit No. 8, Los Gatos, CA 95032, phone (800) 268-6082, fax (408) 356-9570. — *R. Eckhouse*

**Reader Service 28**

## Top-of-the-line scanner

One of the little pleasures of being a reviewer is that you often have a chance to try products as a result of having reviewed other products. Such was the case with Paper Keyboard, reviewed in this issue, which utilizes the Hewlett-Packard Scan Jet Plus for input. I have often heard how well others like the Scan Jet, but I have never had the opportunity to actually use it. Having now done so, I can say I am very impressed and have really come to appreciate the HP quality embodied in this fine product.

Outwardly there doesn't seem to be much to a scanner. The unit is relatively compact (being smaller than either of my printers) and has no knobs or buttons except for an on/off switch. It simply attaches to an interface card that you insert into an empty slot in your computer. Since the Scan Jet Plus can be attached to either a Mac or a PC, the interface kit is sold separately and comes complete with the necessary cable. In my case, I attached the scanner to a PC by following the detailed and illustrated manual that comes with the interface card. Having done so, I next installed the software that comes with the scanner, checked out my installation with the test target and Scantest software that come with the unit, and immediately used the system with the Paper Keyboard software. Everything worked perfectly.

But there is a lot more to this package than just using it as an accessory. The software that comes with the scanner actually includes two separate pieces, Scanning Gallery Plus 5.0 and HP Paintbrush. Both are installed into Windows (versions 2.11 or 3.0) and operate as window applications. Essentially, the Scanning Gallery package is the software equivalent of the knobs and buttons I spoke of earlier. With this software you can set the image type (line art, halftones, diffusion, and gray scales), the resolution, scaling, brightness and white/black levels, and so on, to produce the best image.

Menu selections across the top include zooming, printing, effects, and options. The effects menu includes mirror and negative selections, while the printing menu options produce a page that either demonstrates different exposure settings or halftone image samples. Judging by some images I've obtained with other scanners, the Scan Jet Plus wins hands down in the quality of reproduction — and that's using the automatic setting. I probably could have done even better if I had manually adjusted some the options.

Scanned images can be saved in a number of formats, including TIFF, PCX, EPS, MSP, and IMG. The images can be saved as a file or passed to HP Paintbrush for further work. HP Paintbrush is a lot like the PC Paintbrush that comes with Windows 3.0, only it's much better for several reasons. Right off the bat, it has a better human interface, even though it is only subtly changed (like setting the line width) from that found in PC Paintbrush. HP Paintbrush is also much more versatile because it includes

- more menu options (like filtering an image);
- control of font characteristics (such as gradient, shadow, and intercharacter and interline spacing); and
- support for 256 colors.

Since the software is included free of charge, it's a really nice bonus.

In all ways, the scanner and the software are top rate. About the only other thing worth mentioning is that the otherwise excellent manual is a bit terse when discussing the detail of HP Paintbrush. Surprisingly enough, nowhere could I find a summary of the characteristics of the scanner (that is, the number of gray scales, the maximum size of an image to be scanned, and extra cost options like the sheet feeder).

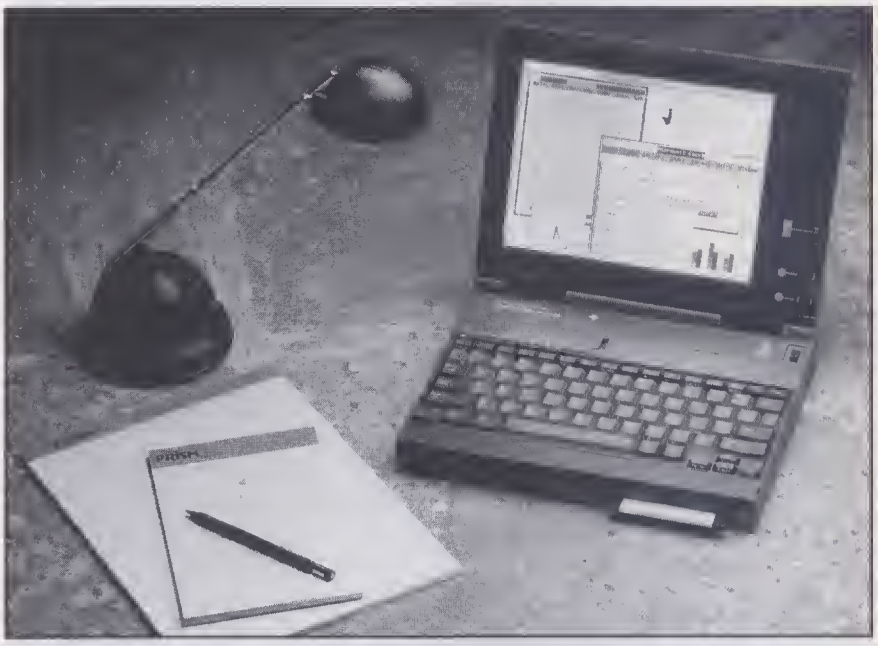
As many readers know, the Scan Jet Plus is "the" scanner most often supported by other software vendors, which makes it easy to justify owning one. In my case, there is no question about wanting to own this scanner. But quality doesn't come cheap. The list price for the scanner is $1,595, with the interface separately priced at $595 for either the Mac or the PC. Fortunately, considerably lower street prices can make this highly recommended scanner a product of choice.

Contact Hewlett-Packard, 700 71st Ave., Greeley, CO 80634, (303) 350-4687. — *R. Eckhouse*

**Reader Service 29**

# NEW PRODUCTS

## Notebook computer features the 386


Texas Instruments Travel Mate 3000 notebook computer.

The Texas Instruments Travel Mate 3000 notebook PC features a 20-MHz 386SX processor; a 10-inch-diagonal, black-and-white VGA display; and a 20- or 40-Mbyte hard-disk drive. The unit weighs 5.7 lbs. with battery and measures 8.5 × 11 × 1.8 inches. An internal 1.44-Mbyte drive uses 3.5-inch floppy disks.

The computer keyboard contains 79 keys with an embedded numeric keypad and a dedicated cursor control pad. Users may run Windows or other graphics applications on the display, which has a resolution of 640 × 480 and 32 gray scales.

Travel Mate comes with MS-DOS V. 4.01, the Laplink data-exchange program, and battery checking and conservation programs. A removable, rechargeable Nicad battery powers the system for about three hours; an AC adapter/recharger is also included.

A Centronics-type parallel interface and an RS-232 serial port provide communications. Other interfaces are provided for an external monitor, a PS/2 mouse connector, and an external numeric keypad. Options include a 2,400-baud modem that can send faxes, an 8037SX numeric coprocessor, and expandable RAM.

Suggested list price for the Travel Mate 3000 is $5,499 for the 20-Mbyte hard-disk-drive model and $5,999 for the 40-Mbyte version.

**Reader Service 30**

## Desktop PC has i486 power

The Bravo 486/25 PC from AST Research reputedly runs nearly twice as fast as 33-MHz 386 systems.

The PC's integrated board includes logic, up to 16 Mbytes of memory expansion, Super VGA graphics (800 × 600 resolution), and an integrated drive electronics interface connector. Weitek coprocessor support, one parallel port, two serial ports, and five full-size, 16-bit expansion slots are also provided.

A standard system includes 2 Mbytes of memory, AST MS-DOS 3.3, built-in password security, and support for four drive bays.

The chassis measures 15.5 × 6.25 × 16 inches deep. The front panel has a power switch, reset button, speed indicator, and chassis lock.

Prices are from $3,995 to $5,365, depending on the disk drives selected and memory capacity.

**Reader Service 31**

## Workstations offer extended memory, upgrade options

Reply Corp. has developed a series of 32-bit Micro Channel architecture workstations and extended memory options. Models 386/25, 25C, and 33C, and models 486/25 and 33 are built around a Turbo Processor module that contains microprocessor and math coprocessor sockets with associated logic. Systems can be upgraded by changing the Turbo Processor module. The 486/25 and 486/33 models contain an 8-Kbyte cache and a math coprocessor.

The overall design includes five disk-drive bays and five full-length expansion slots that accept standard Micro Channel option cards. The system board integrates a 4-Mbyte system memory (expandable to 16 Mbytes), extended VGA graphics, two parallel and two serial ports, and keyboard and pointing-device ports.

Prices range from $3,995 for a 386/25 system to $12,895 for the 486/33.

**Reader Service 32**

## Computers targeted for engineering and real-time applications

**Meiko's Computing Surface systems support multiple users running multiple tasks and permit processors to be assigned to specific tasks.**

Meiko World's two new Computing Surface systems are scalable multiprocessor, multiuser computers for engineering/scientific and real-time applications. The parallel processor systems support from two to thousands of processors that may come from different vendors yet operate together, allowing users to match a job to the type of processor best suited for the task.

The two systems are called the Engineer's Computing Surface and the Embedded Real-Time Computing Surface. The Engineer's Computing Surface is designed for application developers working on simulation, modeling, design, analysis, and other engineering and scientific applications. A typical entry-level system consists of four Inmos T800 Transputer central processors, 1 Mbyte per processor of memory, eight expansion slots, Surfaceware parallel development software, and a Fortran or C compiler.

The Embedded Real-Time Computing Surface is designed to be part of a larger deployed system for applications such as command, control, and communications/intelligence, and radar and sonar image analysis, data acquisition, and industrial control. An entry-level system includes two Intel i860 processors, each with 4 Mbytes of memory; eight expansion slots; and targetable Surfaceware development software.

The company's Surfaceware software is designed to enable programmers to write application code in Fortran or C in a familiar Unix environment. Surfaceware builds parallel programs on the "communicating sequential processes" model, which lets programmers treat an application as a set of ordinary sequential routines that exchange data through Surfaceware's message-passing library routines. A debugger for single- and multiprocessor applications is included.

The entry-level systems cost $35,000 for the Embedded Real-Time Computing Surface and $50,000 for the Engineer's system.

**Reader Service 33**

## Sparc-based servers support Unix System V. 4.0

International Computers' DRS 6000 servers combine Sparc RISC technology with a symmetrical multiprocessing operating system. The series features a 33-MHz Sparc chip with an on-board floating-point coprocessor.

Two servers are available: dual processor and four processor. Both provide symmetric shared-memory multiprocessing across networks with many users and mixed work loads.

The series features Posix and X Open Portability Guide Issue 3 interface standards, Sparc Applications Binary Interface compliance, and support for standard networking facilities.

The DRS 6000 servers include dual-bus and multiple-cache architectures and hardware cache coherency. They use a 40-Mbyte-per-second VMEbus for I/O and a 133-Mbyte-per-second high-speed private bus for CPU and memory traffic. According to the company, the 128-Kbyte write-back memory caches enable CPUs to run at maximum clock speed and minimize contention for the HSP bus. A 64-Kbyte cache on the central services module provides a data path for VMEbus-initiated transfers to memory. The hardware bus-watching logic ensures date integrity in multiple caches.

The servers also feature up to 128 Mbytes of main memory, support for up to 19 Gbytes of SCSI unformatted disk storage, and 31 VMEbus expansion slots for I/O controllers.

DRS 6000 pricing starts at $150,000.

**Reader Service 34**

## WORM disk stores 7 Gbytes of data

Maxell Corp.'s 12-inch, 7-Gbyte optical storage disk features a 1.5-micron track pitch and the zoned constant angular velocity recording method. The OC321-2, designed to operate with Hitachi's OD-321, runs at 1,000 rpm and transfers data at 2.2 Mbytes/ per second.

The product features a pit-edge detection method of recording that increases the disk-pit capacity 1.7 times over previous approaches. The company plans to begin shipment in the first quarter of 1991 at a cost of $650.

**Reader Service 35**

## Smalltalk V works with Windows

Digitalk announced Smalltalk V Windows, an object-oriented programming environment combined with Microsoft Windows 3.0. The company says that Smalltalk V simplifies graphic environment subsystems by providing classes that hide details.

In addition to original Smalltalk V features, such as browsers, inspectors, and push-button debuggers, the product includes interfaces to Dynamic Data Exchange, which allows information sharing with other programs. Also, Dynamic Link Libraries provide a calling application mechanism outside Smalltalk V.

Smalltalk V costs $499.95 and comes with electronic support on Compuserve.

**Reader Service 36**

## Windows available for Unix-based platforms

Ingres Corp.'s Windows 4GL, a visual programming tool and fourth-generation language development system, runs on Hewlett-Packard HP-UX, IBM RISC 6000, DEC Ultrix/Ultrix SQL, and the SCO's Open Desktop. Windows 4GL, with a Macintosh-like graphical user interface, allows users to visually build and modify database applications by selecting elements with a mouse and arranging them on screen.

According to the company, the product's object-oriented fourth-generation language decreases the number of lines of code that need to be written to build an application by as much as 90 percent. The life-cycle management provides data dictionary control over application elements and allows several users to work simultaneously on multiple versions of the same application.

For two to eight workstations, Windows 4GL costs $1,400 per node.

**Reader Service 37**

## Color workstation operates in IBM environments

Decision Data's 3697 workstation functions in IBM System 34, 36, 38, or AS 400 environments in both 80- and 132-column applications. It emulates IBM 5292, 3197C, and 3197D terminals and IBM 5219 and 3812 printers. The workstation also emulates the Hewlett-Packard Laserjet II and IBM Quickwriter.

The workstation features a 14-inch, seven-color Super VGA monitor, keyboard reprogrammability, and an on-screen interactive calculator. When emulating the 3197C or 3197D, the workstation features two or three terminal sessions and one system-addressed printer session. The 3697 also features two simultaneous emulations for a 3197C session, a 3197D session, and a printer session. Users can send printer commands directly from the workstation to the printer through interactive displays and menu-driven setup screens.

Horizontal and vertical split screens allow simultaneous scroll, jump, and zoom control. A printer driver cartridge extends the attached printer list by uploading a driver from the cartridge to the workstation.

**Reader Service 38**

**Screen print with trim and page-positioning capabilities are standard with the 3697 color workstation from Decision Data.**

**Portable fax holds about 30 letter-size sheets.**

## Compact fax sends standard documents

Ricoh's 5.5-lb, portable fax machine can send and receive letter-size documents or serve as a copier. The 11 × 7 × 2-in. PF-1 connects to phone lines via a standard RJ11C jack or runs from a battery pack.

The PF-1 error correction mode reputedly corrects transmission communication errors without requiring user retransmission. ECM checks incoming document frames and demands automatic retransmission of error-filled blocks. This mode also allows communication with models from different manufacturers.

The PF-1 transmits documents at 34 seconds per page and 4,800 bits per second in two resolutions, 100 × 200 and 200 × 200 lines per inch.

Standard PF-1 accessories include a car adapter for the cigarette lighter, an AC adapter for standard wall outlets, and the battery pack and charger. Options include a carrying case, an acoustic coupler, and a spare battery pack. Basic price is $1,695.

**Reader Service 39**

## Operating system opens up computing environments

The Open Software Foundation has announced the first release of the OSF/1 operating system. It is compatible with Unix System V and Berkeley programming interfaces and can be used in the Intel 302, Digital Equipment Corp.'s Decstation 3100 workstation, and the Encore Multimax multiprocessor system, among others.

OSF/1 supports the OSF/Motif graphical user interface and employs a Mach-based kernel from Carnegie Mellon University that allows workloads to be distributed among multiple processors. The kernel offers dynamic system configuration, logical volume management, and disk mirroring.

Other integrated technologies include portions of IBM's Aix V. 3.1 operating system, System V and BSD4.3 commands, and the Berkeley 4.4 Virtual File System.

OSF plans to release subsequent versions of the system every 12 to 18 months, culminating in microkernel implementations.

**Reader Service 40**

## Image scanner available on HP Unix workstations

Tetra Systems' Tetra Scan image scanner system is now available for the HP 9000-series 300 and 400 workstations. The system provides an OSF-Motif-based application interface to scanner products such as the HP Scanjet, Ricoh IS-30, and Howtek Scanmaster. A graphical user interface allows interactive selection of the scanned area while providing control over image resolution, quantization, and halftones. A parallel scanner interface board comes with the system.

An optional $3,500 image-scanning coprocessor provides a bidirectional Centronics port and a dedicated 10-MIPS coprocessor with 2 Mbytes of image memory. The network-transparent user interface allows several workstations to share one scanner.

Tetra Scan can save images in a variety of file formats, including TIFF, PCX, TARGA, and ISP. These images can be used with other application software or printed on an HP Laserjet or Paintjet.

Tetra Scan costs $2,495. Software-only and accelerated configurations are available from $1,495 to $4,700.

**Reader Service 41**

## Enhanced version of Cobol/2 compiler for Unix announced

Version 1.2 of the Micro Focus Cobol/2 compiler for Unix includes the company's Cobol/2 native-code-generating compiler and programmer productivity tools — such as a visual debugger — in a bundled system for production and maintenance of Cobol applications.

The release is available for AT&T Unix System V. 4 and SCO Unix operating systems on i80386 workstations and for other platforms from computer manufacturers that resell Micro Focus Cobol/2 products.

Features in the enhanced version include indexed sequential access method file data compression and file key compression, and support for color, wide terminals, and attached printers. Another extension supports the setting of dynamic screen attributes from within the user program and provides compatibility with RM/Cobol-85.

**Reader Service 42**

## Alsys introduces Ada for Macintosh

Alsys's software engineering environment for the Macintosh contains a production-quality Ada compiler, a menu-driven interface, and a set of programming tools to develop Ada applications.

The $1,815 environment runs under the Macintosh window-based programmer's workshop, which allows developers to perform program editing, file manipulation, compilation, and program execution. The applications can run either as MPW tools or as stand-alone applications.

Alsys allows users to write Macintosh-like applications through Ada interface packages to the Macintosh toolbox. Existing code written in other languages may also be reused through interfaces to MPW C and Pascal.

The Alsys compiler includes high- and low-level optimizers, and the library environment contains family, library, and unit managers. The toolset includes a symbolic source-level debugger and program viewer, a cross-reference generator, a recompilation aid, and a source reformatter. The software features a runtime executive, a standard user interface, required Ada packages, and an ISO-standard math library.

The product runs on 68020- or 68030- Apple Macintosh computers with system software V. 6.0 or above and generates code for any Macintosh computer. Four Mbytes of main memory (8 Mbytes with Multifinder) and 15 Mbytes of disk space on a single disk volume are required.

**Reader Service 43**

## Hewlett-Packard introduces C++ compiler

Hewlett-Packard now offers a C++ compiler based on AT&T C++ V. 2.1 on the HP-UX operating system. The company describes it as a "true" C++ compiler, meaning that it generates object code directly from C++ source code. The new release increases compile-time performance on HP-UX operating systems up to 75 percent, according to the company.

HP's C++ version 2.1 compiler costs $1,700.

**Reader Service 44**

## Real-time compression technology runs on DOS systems in several forms

Stac Electronics' Stacker enables IBM PCs and compatibles to double Winchester disk storage without a new hard drive. The product is available in three forms: in a 30-Kbyte software program for laptop, notebook, and Micro Channel computers; with an add-in board for IBM PCs and compatibles; and in coprocessors for new systems.

Stacker provides continuous compression and decompression (transparent to the user) without data loss or user intervention during read and write requests to the disk drive. Using a compression ratio of 2:1, Stacker reduces most PC data to half its former volume. In some cases, the company claims, compression ratios can reach 15:1.

The product is compatible with MS-DOS and PC-DOS 3.x and 4.x, including Compaq's 3.31 DOS version. Stacker works with operating environments such as Microsoft Windows 3.0, DOS commands, disk-caching programs, utility programs, and all hard drives.

A device driver development kit is available to OEMs that features device driver object code, sample hardware interface source code, hardware interface software specification, a driver validation test suite, and two evaluation boards.

The software-only version costs $129; with a coprocessor board, Stacker costs $229.

**Reader Service 45**

UNC CHARLOTTE

**CHAIRPERSON**
***Department of Computer Science***
**COLLEGE OF ENGINEERING**
*Search Continued*

Applications and nominations are invited for the position of **Chairperson of the Department of Computer Science**. Applicants must have a Ph.D. or equivalent in Computer Science or a related field and must show a successful record of research in computer science, computer engineering, or information science. In addition, the individual must have a strong interest in teaching and research at both undergraduate and graduate levels, and exhibit academic and administrative leadership qualities. The position will be at the rank of Professor with a highly competitive salary. **Anticipated starting date is July 1, 1991.**

UNC Charlotte is one of the largest institutions of the UNC System. It has more than 14,000 students including 2,100 graduate students in the six colleges of Arts & Sciences, Architecture, Business Administration, Education & Allied Professions, Engineering, and Nursing, and in the Graduate School.

The Department of Computer Science has 24 faculty and is the largest of the five departments within the College of Engineering. It offers a B.A., a B.S., and an M.S. degree in Computer Science and over the next few years will continue the development of its research and graduate programs including doctoral level work. The university if firmly committed to providing personnel and facilities for this department including participation in a new 75,000 sq. ft. Applied Research Center. Our immediate proximity to the University Research Park with tenants such as IBM, Verbatim, AT&T, Bell South, etc. and our participation in the Microelectronics Center of North Carolina, the North Carolina Supercomputing Center, and the Engineering Research Center greatly enhance our education and research activities. Current faculty strengths are in the areas of artificial intelligence, computer engineering, database systems,theoretical computer science, and computer networks.

With a metropolitan population of over one million, Charlotte is the largest city in the Carolinas. Located within a few hours drive of the mountains and the ocean, Charlotte has a moderate climate, attractive neighborhoods, and a multitude of cultural and recreational opportunities. Charlotte Douglas International Airport is one of the busiest in the South and Charlotte is among the largest banking centers in the United States.

Nominations and letters of application including a resume and names of four references should be addressed to: **Dr. Robert Carrubba, Chairperson, CSCI Search Committee, College of Engineering, UNC Charlotte, Charlotte, NC 28223.** Initial screening is underway, and early submission of applications is encouraged, although applications will be accepted until the position is filled. AA/EOE

# IC Announcements

| Company, Model, Function | Comments | R.S. No. |
|---|---|---|
| Advanced Micro Devices<br>PAL22V10<br>PAL device | Mil-Std-883C 12-ns version of the PAL22V10 for DSP or avionics operates at 50-MHz maximum frequency. Also available in 10-ns version. Macrocell allows design flexibility. Comes in 24-pin ceramic DIPs, flatpacks, and 28-pin LCCs. Cost (100s): $57.90 (ceramic DIPs); $104.25 (flatpacks and LCCs). | 120 |
| Analog Devices<br>AD9620/AD9630<br>Buffer amplifiers | Unity-gain wide-band buffer amplifiers with closed-loop architectures. The AD9620 features 0.989V/V minimum gain accuracy over temperature for a 2V peak-to-peak input swing. The AD9630 has a minimum gain accuracy of 0.983V/V. Slew rates are 2,200- and 1,200V/μs with 1.6- and 1.5-ns maximum rise-and-fall times for a 1V step over the operating temperature ranges. Both devices feature an output stage to minimize series resistance. Cost (100s): $19, AD9620; $6.25, AD9630. | 121 |
| Cirrus Logic<br>CL-CD2401<br>Intelligent controller | Four-channel multiprotocol communications controller serves as intelligent coprocessor. Combines with eight-channel DMA controller on one chip. Meets US, European, and Japanese modem-control signal standards. Receives and monitors data from different-protocol channels and sends or responds to flow-control characters. Features embedded RISC processor that offloads functions from CPU. Cost (1,000s): $31. | 122 |
| Hitachi<br>HA13481S/HA13501S<br>Drivers-controllers | Intelligent spindle-motor driver/controllers for hard drives that do not require Hall-effect sensors. The HA13481S runs 3.5-inch drives on 12V; the HA13501S controls 2.5-inch and smaller drives on 5V. Features discriminator circuitry for programming spindle speeds by selecting external oscillator frequency, divide ratio, and count number. Cost (50,000s): $3.40, HA13481S; no price given for HA13501S. | 123 |
| Integrated Device Technology<br>IDTR3051/IDTR3052<br>Controllers with chipset | Meet JIAWG military standard for 32-bit avionics systems. One-chip CPUs feature integrated MIPS R3000A processor, instruction and data cache, memory management unit, clock generator, DMA arbiter, and four deep read and write buffers. CPUs rated from 16 to 35 MIPS at 20- to 40-MHz operation. MMU comes with optional translation lookaside buffer. Cost (10,000s): $30, R3051 (6-Kbyte cache); $49, R3052 (10-Kbyte cache). | 124 |
| International CMOS Technology<br>PA7024P-1/PA7024J-1<br>PEEL arrays | Programmable, electrically erasable logic arrays support 60-MHz system clock rates and furnish 20 I/Os, two input/system clocks, and 20 buried logic control cells. Each can be configured to have D, T, or JK registers with independent or global clocks, presets, and resets. Cost (1,000s): $16.67, PA7024P-1; $17.42, PA7024J-1. | 125 |
| Micro Linear<br>ML4819<br>Power-supply controller | Combines power-factor corrections and pulse-width modulation into single device. Reduces control-section size in power supplies for PCs in 150- to 400-watt range, as well as for computer peripherals, plotters, and printers. Cost (100s): $3.95 (20-pin DIPs). | 126 |
| Motorola<br>68HC05E1<br>CSIC microcontroller | Software-programmable phase-lock loop oscillator subsystem, a real-time clock-interrupt circuit, and 364-byte RAM storage with HC05 CPU chassis. Features 4-Kbyte ROM, a 32-KHz PLL oscillator, a 15-stage timer, and 20 I/Os. Originally designed for Macintosh LC and IIsi. Cost (depending on volume): $3 to $3.50 . | 127 |
| NMB Technologies<br>AAA1M300 series<br>DRAMs | CMOS devices come in 1-Mbit × 1 and 256-Kbit × 4 versions. Feature 53-, 60-, and 70-ns maximum access times and 100-ns read/write cycle time. Allow direct memory access with 16- and 20-MHz processors. Enhanced page mode available with 40-ns read/write cycle time. Requires power consumption of 400 mW. Cost (10,000s): $5. | 128 |
| SGS-Thomson Microelectronics<br>ST5421<br>S-transceiver | CCITT I.430-compliant IC handles four-wire S-interface at 192 Kbits/sec., carrying two B channels at 64 Kbits/sec. each for data and voice transmission. Also carries D channel at 16 Kbits/sec. for signaling and packet data transfer. Uses general circuit interface for communication. No price given. | 129 |
| Twinhead<br>TH4100<br>AT chip | One-micrometer ASIC comes in 208-pin quad flatpack with a gate count of 20,500 (80,000+ transistors). Chip implementation of IBM PC AT system logic includes system bus controller/buffer, integrated peripherals controller, and EMS memory controller. Supports 80286 and 80386SX CPUs at 12, 16, or 20 MHz. No price given. | 130 |

# Microsystem Announcements

| Company, Model, Function | Comments | R.S. No. |
|---|---|---|
| Doctor Design XQC-8200 X Windows controller | Supports color monitor resolutions of 1,024 × 768, 800 × 600, and 640 × 480 at 8 bits per pixel. Provides 256 colors from a 16-million-color palette. Levels of DRAM expansion available from 2 to 24 Mbytes with 1-Mbyte VRAM for image memory. ROM expands to 2 Mbytes for system boot, hard-loading X server software, and font storage. Cost: $75,000 (OEM licensing; hardware royalties from $10). | 135 |
| ICS Electronics ICS 2323A Serial interface | Supports computers or other serial sources that require a parallel data word. Translates serial messages into latched BCD/hex or binary data or vice versa. Parallel interface configured for 10-BCD/hex input and 10-BCD/hex latched output. Fits most applications with plug-in EPROM, operates with RS-232/422/485 ports, and accepts asynchronous formats and baud rates. Cost: $360. | 136 |
| Klever Computers K386-25/33 Small AT motherboard | Features 128-Kbyte on-board cache with optional SRAM, up to 16 Mbytes of 80-ns SIMM, and eight expansion slots. Runs on 25- or 33-MHz Intel 386 processor and includes 80387 coprocessor socket, Weitek math coprocessor, and shadow RAM for system and video BIOS. Supports six 16-bit AT slots and two 8-bit XT slots. Compatible with OS/2, PC-DOS, Unix, Xenix, and Novell software. Cost: $1,050 (2 to 9 units). | 137 |
| Mercury I/OMax1 Daughterboard | Daughterboard for Mercury MC860 i860-based processor provides Maxbus interface transfer rate of up to 20 Mbytes/sec. Bidirectional FIFO packs 8 or 16 bits to and from 32-bit word. Data transfers to MC860 memory at 80 Mbytes/sec. over DMA interface. Cost: $2,090. | 138 |
| Mesa Electronics 6M22F disk emulator | Rugged, solid-state disk replacement features short-slot (4.2 × 5.5-inch) PC bus card that operates in temperatures from -40° to 85°C. Comes with on-board firmware and software utilities that support directory buffering to minimize EEPROM write cycles. Uses less than 250 mW of power and features eight 32-pin JEDEC standard memory devices. Stores up to 4-Mbytes data; expansion available with additional cards. Cost: $149 (100s). | 139 |
| MNC International QC 9230x LAN cards | Series of Arcnet cards for NEC Prospeed 286, 386SX, and 386 laptops connect to coaxial networks in either star or bus modes. A twisted-wire-pair model is also available. The Novell-compatible cards run on laptop batteries at 2.5 Mbps. Cost: $385, QC 9230-2 (star); $395, QC 9230-5 (bus); $445 QC 9230-6 (twisted-wire pair). | 140 |
| Newer Technology Attraction+ Memory board | Board for ISA and EISA 286, 386, and 486 PCs supplies up to 16 Mbytes of 16-bit memory. Upgradable in increments of 2 Mbytes with 1-Mbyte SIMMs or of 8 Mbytes with 4-Mbyte SIMMs. Supports 12.5-MHz bus speed and 33-MHz CPU operation in DOS, OS/2, Novell, Windows, Unix, and Xenix environments. No cost given. | 141 |
| SBE Vcom-25 Communications controller | Serial interface for OEMs, systems integrators, and VARs developing 9U VMEbus systems and workstations. The 20-MHz, 68020-based board provides four or eight full-duplex, independently programmable ports with T1 speeds up to 1.544 Mbps. Available with X.25 software for wide-area networking. Configures for RS-232C/449 and V.35 requirements. No price given. | 142 |
| Themis Tsvme 551X Network controller | Includes six serial channels with modem support, two Motorola 68302 integrated multiprotocol processors, and a 68020-based concurrent processing architecture. Features 20-Mbyte/sec. VMEbus DMA transfer rate. Software support includes OSI-compatible X.25, PAD, and TCP/IP backplane drivers; Unix, OS-9, and Vrtx velocity drivers also available. Cost: $3,950. | 143 |
| Win Systems MCM-SBC42 Single-board computer | Intelligent board based on the NEC8 or 10-MHz V40 processor operates independently of master STDbus CPU or as control coprocessor. A 32-Kbyte shared memory mapped into the main system communicates with master STDbus processor. Provides synchronous or asynchronous data communication over two serial I/O ports. ROM firmware supports DOS and embedded systems applications. Consumes less than 5W and requires +5VDC. Cost: $595. | 144 |
| Zeos International Zeos Notebook 286 Lightweight PC | Weighs less than seven pounds and includes a 1-Mbyte memory, VGA display, 20-Mbyte hard disk, and 1.44-Mbyte floppy drive. Runs at 12.5 MHz for about two hours on a snap-in battery. Features an 80286 processor, 82-pad keyboard, and standard I/O port connectors for peripherals. Cost: $1,995. | 145 |

Editor: Edmund L. Gallizzi, Computer Science Department, Eckerd College, St. Petersburg, FL 33733, phone (813) 864-8272, e-mail e.gallizzi@compmail.com

# Honoree Dijkstra says students should challenge popular trends

*Anish Arora and Paul C. Attie*
*University of Texas at Austin and Microelectronics and Computer Technology Corp. (MCC)*

Speaking during a two-day symposium honoring his 60th birthday, Edsger W. Dijkstra of the University of Texas at Austin asserted that the most important thing academia can teach its graduate students is to avoid jumping on bandwagons. Instead, he said, students should subscribe only to those concepts that they have good technical reasons to believe in.

During his evening banquet talk May 10 in Austin, Dijkstra related some of the major milestones of his 39-year computer science career, including the implementation of Algol 60 and his work on process synchronization and program derivation. The honoree thanked the Burroughs Corporation for appointing him a research fellow, noting that he had found it hard to penetrate industry even while working for a company. He also acknowledged the UT/Austin Department of Computer Sciences and the symposium organizers, Robert Boyer, Jayadev Misra, and Hamilton Richards.

The main speaker at the banquet, David Gries of Cornell University, praised Dijkstra's discipline, intense desire to be honest, and skill with the pen that most people would "kill to have." A number of Dijkstra's associates made short speeches. Three letters of tribute were read, one from the Polish Information Processing Society, another from Brian Randell of the University of Newcastle upon Tyne, and a third from Don Brabin of British Petroleum Venture Research.

**Frontiers-in-computing theme.** The organizers called the May 10-11 symposium "Frontiers of Computing — A Tribute to Edsger W. Dijkstra." It featured eight invited presentations.

Mani Chandy of the California Institute of Technology opened the proceedings with a presentation on "Simultaneity in Distributed Systems" in which he addressed the inadequacy of intuitive reasoning for establishing correctness of distributed systems. He noted that operational arguments about the behavior of distributed systems are complex since components in a distributed system operate concurrently and at possibly different speeds. In using intuitive arguments, he said, you risk overlooking some scenarios.

Chandy illustrated this view by conjecturing a method to check, in a piecemeal fashion, whether actions occurring in different components are simultaneous. To establish correctness of the method, he first showed a simple, two-dimensional geometrical proof for the case in which each communication channel in the distributed system is accessed by at most two components. He next presented an intuitively plausible argument based on diagrams, which implied that the simple proof continues to hold when generalized to $n$ dimensions ($n>2$), thereby accommodating the general case in which each communication channel is accessed by up to $n$ components.

On careful consideration, however, it turned out that the generalization failed for certain intricate sequences of actions that were not obvious in the diagrammatic argument. Chandy concluded, "While intuition is often helpful in formulating solutions, correctness is best established by formal reasoning."

**Mechanical verification.** Jay Moore of Computational Logic spoke on "An Applicative Theorem Prover Written in its Own Logic, The Saga Continues," setting forth the view that the main goal of mechanical theorem-proving is to make the verification of programs practical.

Moore related his work to Dijkstra's by stating, "Mechanical methods will not be practical until manual methods (that is, proofs by hand) are." Therefore, Dijkstra is doing all of us "great service" by his work on manual methods. He cited the recursive unsolvability of the halting problem and Gödel's Incompleteness Theorem as examples of fundamental theorems that have been proven by the Boyer-Moore theorem prover.

Moore's vision of the work is that verification is needed for all levels of a system: the runtime environment, high-level language compiler, assembler, and linking loader as well as the application program. Computational Logic has designed such a system and verified each level using the Boyer-Moore theorem prover. Moore illustrated the approach by giving an example of a small program and its code listing at each level of this system. He stressed the need to be concerned with verification down to the bit level and that, while source program correctness is important, it is only the "tip of the iceberg." A major new project (the source of the title of the talk) is a new theorem prover that supports a subset of applicative Common Lisp.

**Nondeterministic programs.** Misra of UT/Austin spoke on "Equational Reasoning about Nondeterministic Programs," commenting that an alternative title could be "Nondeterminism Considered Harmful." He first reviewed the equational technique of Kahn, which describes a network of deterministic communicating processes by a set of equations. An equation describes the output of a process as a function of its inputs, and the set of equations can be solved algebraically. Interestingly, the solution of these equations is the sequence of messages that flows along each channel.

Misra said that in distributed systems many of the component processes are inherently nondeterministic; in some states, these systems may execute any of several actions. One such example is a fair merge process, which has two input channels and a single output channel. A message arriving on either of the input channels is guaranteed to be eventually placed on the output channel.

Nondeterminism implies that a given output sequence is not uniquely determined by the input sequence; in the fair merge, for example, there are many output sequences that would be the fair merge of a given pair of input sequences. This phenomenon prevents the straightforward application of Kahn's method to networks containing nondeterministic processes, since it causes the set of equations to have multiple solutions, some of which are nonsensical in that they violate the normal laws of cause and effect.

In the remainder of his talk, Misra outlined a technique for modifying Kahn's method to eliminate the nonsensical solutions.

**Proof design and program derivation.** Nettie van Gasteren of the University of Utrecht and Wim Feijen of the Eindhoven University of Technology, two institutes in the Netherlands, jointly "simulated" a typical session of the Tuesday Afternoon Club. (The club is a discussion group that Dijkstra organized first at Eindhoven and later at UT/Austin, where it currently meets.) Their simulation consisted of two parts, one on proof design and the other on program construction, both of which are frequent themes of club meetings.

Van Gasteren suggested that the design of calculational proofs can be reasonably based on the "shape" of the formulae in the proofs. She illustrated this claim using an example in which the design decisions at each proof step were based on heuristic techniques similar to the ones adopted and developed in the recent book, *Predicate Calculus and Program Semantics* (Springer-Verlag, New York, 1990), coauthored by Dijkstra and Carel Scholten.

Feijen continued the session with the claim that programs can be reasonably designed hand in hand with their correctness proofs. As an example, he considered the problem of determining, in linear time, the lexicographic minimum of a circular array. Feijen conjectured that a single Do-loop would suffice for solving the problem in linear time, and designed the invariant accordingly. Once the invariant had been completely determined, he called upon van Gasteren to extemporaneously complete the program, which she did with remarkable ease. He then gave a simple termination argument to complete the program design.

**Algebra of lists.** David Turner of the University of Kent wrapped up the first-day's proceedings with a study in the algebra of lists. His talk, "Duality and De Morgan Principles for Lists," drew inspiration from Dijkstra's belief that "a study in the algebra of Boolean operations should not be taken trivially."

Based on the observation that the number of useful list-processing operations is quite large, Turner emphasized the need for a set of principles to organize the algebra of lists. He suggested that such an organization is made possible by the insight that, for finite lists, the list-reverse operation plays a role analogous to that of negation in the Boolean algebra. Thus, the list analog of the duality principle in Boolean algebra is that each algebraic identity over finite lists remains true if every list operation is replaced by its dual.

By analogy to the De Morgan principle, $REV\ (f\ x_0, x_1, ..., x_N) = g\ (REV\ x_0)\ (REV\ x_1)\ ...\ (REV\ x_N)$, where $REV$ is a context-dependent generalization of the list reverse operation, and $f$ and $g$ are dual operations that take $x_0, x_1, ..., x_N$ as arguments.

Turner said that a methodological implication of this work is that if an operation is not its own dual, then its dual operation should be named. Also, the fact that these organizing principles work only for finite lists is evidence that finite and infinite lists should be treated as different types.

**Program extensibility.** Niklaus Wirth of ETH Zurich, a long-time associate of Dijkstra, opened the next day's session with a talk entitled "Program Extensibility." He focused on the object-oriented programming paradigm and asserted, "It is dangerous to come up with a completely new paradigm that is the silver bullet — a solution to all your problems. I prefer to work in an evolutionary way."

Wirth went on to state that separation of concerns is important to us, crediting Dijkstra for his contribution in this regard. In the early 1980s, we achieved separation of concerns by dividing programs into modules with well-defined interfaces, Wirth said. In the late 80s, we recognized that not only do we write programs, but we "grow" them.

It is impractical to completely rewrite the code every time a request for more functionality arises, he said. It is also the case that specifications are initially incomplete and, therefore, new requests for functionality arise while the system is in use. Current techniques are inadequate for this, and this is where object-oriented programming can help us, said Wirth. He went on to give an example illustrating how new (sub)classes can be added to a system without requiring extensive changes to already present modules.

**Specifying behaviors.** Wlad Turski of the Warsaw University followed with "Is Specifying Behaviors Similar to Specifying Computations?" dealing with the question of whether general information-processing tasks (behaviors) can be specified in the same manner as sequential computations.

Turski presented two principles to be used in the specification of concurrent systems: (1) A process should not be allowed access to more information than is strictly necessary for it to perform its task, and (2) An unbounded sequence of bad scheduling choices, which causes a request for resources to be denied forever, can be ignored.

The first principle is needed because excess knowledge allows processes to monopolize resources, thereby causing requests for these resources by third parties to be denied forever. Turski argued that the second principle is justified because an unbounded sequence of bad scheduling choices will not occur in practice: it violates the second law of thermodynamics.

Turski went on to present a specification of the dining-philosophers problem that used both of the above principles. He proved that the specification was deadlock free and that a philosopher is prevented from eating only by an unbounded sequence of bad scheduling choices.

At the conclusion of the talk, Doug McIlroy of Bell Laboratories asked, "What if the philosophers resonate with each other and therefore starve?" Turski responded, "In the physical world, no two components will resonate on the same frequency unless stimulated." This was followed by a remark that "axiomatizing an analogue to the second law of thermodynamics for digital systems leads to fairness," and Turski agreed.

**Program semantics.** Jan van de Snepscheut of Caltech gave the final presentation. He recalled one of Dijkstra's seminal contributions, namely, the development of the semantics of guarded command programs using the functions *wp*, the weakest precondition, and *wlp*, the weakest liberal precondition. The functions *wp* and *wlp*, also called predicate transformers, are useful in specifying invariants and capturing a program's input-output relation. Thus, they form a basis for verifying properties of programs.

Snepscheut extended previous work by formulating predicate transformers that capture progress properties of the form $P$ leads-to $Q$. (Informally, $P$ leads-to $Q$ in a program means that if $P$ ever holds in an execution of the program then either $Q$ holds at that point or $Q$ holds eventually.) In particular, he defined four predicate transformers: *wev*, *wlev*, *wto*, and *wlto*. He went on to analyze the characteristics of these transformers (for example, their conjunctivity and disjunctivity) and stated theorems with which loop invariants can be proven using these transformers.

After the talk, Mohamed Gouda of UT/Austin asked whether the predicate transformer approach could be extended to include parallel composition. Snepscheut replied that he had studied this important problem and found it to be a technically challenging one.

**Finale.** At the conclusion of the banquet, Dijkstra was presented a copy of *Beauty is Our Business* (Springer-Verlag, New York, 1990), a collection of 54 short, technical articles contributed by more than 60 of Dijkstra's colleagues and edited (without Dijkstra's knowledge) by Gries, Misra, Feijen, and van Gasteren.

# CALL FOR PAPERS

*IEEE Software* seeks articles for publication in 1991 and 1992 on the following topics: software renovation, work-group computing, maintenance under the object-oriented paradigm, postmortem analysis of software projects (from both technical and management perspectives), embedded systems programming and development, industrial experiences with Ada and C++, human factors issues for software developers, and use of CASE tools in industrial development. Articles in *IEEE Software* focus on results and experience useful to practitioners. Submit eight copies of articles to Carl Chang, *IEEE Software*, 1120 Science and Eng. Offices, MC 154, Univ. of Illinois, Chi-

## Call for papers and referees for *Computer*

*Computer* seeks articles for inclusion in upcoming special issues.

**Distributed Computing Systems** has been selected as the theme for the **August 1991** issue. Prospective authors are invited to submit tutorial, survey, descriptive, case-study, application-oriented, or pedagogic manuscripts. See the July 1990 issue of *Computer* (p. 120) for complete information.

The deadline for complete manuscripts is **January 1, 1991.** Notification of decisions is set no later than **March 15, 1991**, and the final version of each manuscript is due no later than **May 1, 1991**.

Submittals and questions should be directed to either of the guest editors, Mukesh Singhal, Department of Computer and Information Science, Ohio State University, Columbus, OH 43210, phone (614) 292-5839, e-mail singhal@cis.ohio-state.edu; or Thomas L. Casavant, Department of Electrical and Computer Engineering, University of Iowa, Iowa City, IA 52242, phone (319) 335-5953, e-mail tomc@eng.uiowa.edu.

**Multimedia Information Systems** will be the theme of the **October 1991** issue. Manuscripts reporting survey, original research, design and development, and applications of multimedia technology are sought. See the October 1990 issue of *Computer* (p. 100) for complete information.

Eight copies of each complete manuscript must be submitted by **February 1, 1991**. Notification of decision is set for **May 1, 1991**, and the final version of each manuscript must be submitted by **July 1, 1991**.

Submissions and questions should be directed to A. Desai Narasimhalu, Inst. of Systems Science, National University of Singapore, Heng Mui Keng Terrace, Singapore 0511, phone (65) 772-2002, fax (65) 778-2571, e-mail issad@nusvm; or Stavros Christodoulakis, Department of Computer Science, Davis Building, University of Waterloo, Waterloo, Ont., Canada, phone (519) 888-4452 or (30) 821-64846, fax (519) 885-1208 or (30) 821-53571, e-mail schristodoul@waterloo.edu.

**Heterogeneous Distributed Database Systems** is the theme planned for the **December 1991** issue. See the August 1990 issue of *Computer* (p. 115) for complete information.

Abstracts of the manuscripts are due no later than **January 1, 1991**, and eight copies of the complete manuscripts must be submitted by **April 1, 1991**. Notification of decisions is **July 1, 1991**, and the final version of each manuscript is due **September 1, 1991**.

Submissions and questions should be directed to Sudha Ram, Department of Management Information Systems, Eller School of Management, University of Arizona, Tucson, AZ 85721, phone (602) 621-2748, e-mail ram@mis.arizona.edu on Internet or ram@arizmis on Bitnet.

**Parallel Processing for Computer Vision and Image Understanding (CVIU)** is the theme planned for the **February 1992 issue.** Tutorial, survey, architecture case-study, performance evaluation, and other manuscripts are sought.

Subareas of interest include, but are not limited to:

- Architectures: Multiprocessor architectures and special-purpose architectures for CVIU.
- Algorithms: Design, mapping, and implementation of parallel algorithms for CVIU problems.
- Languages: Design of languages for efficient implementation of CVIU programs, especially for parallel processing and architecture-independent implementations.
- Software development tools for parallel CVIU applications.
- Performance evaluation: Benchmarking, performance evaluation of architectures and algorithms; performance evaluation of integrated CVIU systems.
- Real-time vision architectures and applications.

Fourteen (14) copies of each complete manuscript are due by **March 1, 1991**. Notification of decisions is set **August 1, 1991**, and the deadline for the final version of the manuscript is **October 1, 1991**.

Submissions and questions should be directed to Sanjay Ranka, 4-116 Center for Science and Technology, School of Computer Science, Syracuse University, Syracuse, NY 13244, phone (315) 443-4457, e-mail ranka@top.cis.syr.edu; or Alok Choudhary, Department of Electrical and Computer Engineering, 121 Link Hall, Syracuse University, Syracuse, NY 13244, phone (315) 443-4280, e-mail choudhar@fruit.ece.syr.edu.

*For submittal to* Computer, *manuscripts must not have been previously published or currently submitted for publication elsewhere. Each manuscript should be no more than 32 typewritten, double-spaced pages long, including all text, figures, and references. Each submittal should include a cover page that contains the title of the article, the full name(s) and affiliation(s) of the author(s), complete postal and electronic address(es) of all the authors as well as their telephone and fax number(s), a 300-word abstract, and a list of keywords identifying the central issues of the manuscript's contents. The final manuscript should be approximately 8,000 words in length and contain no more than 12 references.*

*If you are willing to review articles for any of these special issues, please send a note listing your research interests to Jon T. Butler, associate technical editor of* Computer, *or to one of the guest editors listed for the particular issue. Butler may be reached at the Department of Electrical and Computer Engineering, Naval Postgraduate School, Code EC/Bu, Monterey, CA 92943-5004, phone (408) 646-3299 or (408) 646-3041, fax (408) 646-2760, e-mail butler@cs.nps.navy.mil.*

cago, IL 60680, Internet ckchang@uicbert.eecs.uic.edu. For detailed author guidelines, contact Karen Potes at (714) 821-8380 or soft.one@compmail.com.

**SCAI 91, Third Scandinavian Conf. on Artificial Intelligence:** May 21-24, 1991, Roskilde, Denmark. Submit three copies of abstract with long or short paper by **Dec. 20, 1990,** to Brian Mayoh, Computer Science Dept., Aarhus Univ., Ny Munkegade, Bldg. 540, DK-8000 Aarhus C, Denmark, phone 45 (86) 127-188, fax 45 (86) 135725, e-mail brian@daimi.aau.dk.

**IEEE Pacific Rim Conf. on Comm., Computers, and Signal Processing:** May 9-10, 1991, Victoria, B.C., Canada. Cosponsors: IEEE Victoria Section, Univ. of Victoria. Submit three copies of summary by **Dec. 20, 1990,** to Technical Program Committee, c/o Pan Agathoklis, Electrical and Computer Eng. Dept., Univ. of Victoria, PO Box 3055, Victoria, B.C., Canada V8W 3P6, phone (604) 721-8618, fax (604) 721-8676.

*Int'l J. of Computer-Aided VLSI Design* plans a special issue on hardware accelerators for CAD. Publisher: Ablex. Submit five copies of full paper by **Dec. 31, 1990,** to Ausif Mahmood, Electrical Eng. Dept., Washington State Univ. at Tri-Cities, 100 Sprout Rd., Richland, WA 99352, phone (509) 375-9234, e-mail mahmood@prime.tricity.wsu.edu.

**Second Int'l Conf. on Algebraic Methodology and Software Technology:** May 22-24, 1991, Iowa City, Iowa. Submit abstract by **Jan. 1, 1991,** to AMAST Conf., Computer Science Dept., Univ. of Iowa, Iowa City, IA 52242.

*IEEE Software* is accepting entries for the fourth annual Gordon Bell Prize competition. Two $1,000 prizes will be awarded. Submit three- to four-page executive summary and full report by **Jan. 2, 1991,** to 1990 Gordon Bell Prize, c/o *IEEE Software*, 10662 Los Vaqueros Cir., PO Box 3014, Los Alamitos, CA 90720-1264, Compmail soft.one, Internet: soft.one@compmail.com.

**Second Physical Design Workshop:** May 20-22, 1991, Laurel Highlands, Pa. Sponsor: ACM SIGDA. Submit 14 copies of the manuscript by **Jan. 7, 1991,** to Mary Jane Irwin, 333 Whitmore Lab, Computer Science Dept., Penn State Univ., University Park, PA 16802, e-mail mji@cs.psu.edu.

**20th Int'l Conf. on Parallel Processing:** Aug. 12-16, 1991, St. Charles, Ill. Sponsor: Pennsylvania State Univ. Submit paper by **Jan. 10, 1991,** to Kimming So, IBM Austin, Internal Zip 2812, 11400 Burnet Rd., Austin, TX 78758 (on algorithms and applications); Herbert D. Schwetman, MCC, 3500 W. Balcones Center Dr., Austin, TX 78759 (on software); and Chuan-Lin Wu, Dept. of Electrical and Computer Eng., Univ. of Texas at Austin, Austin, TX 78712 (on hardware and other subjects).

**Tencon 91, 1991 IEEE Region 10 Conf.:** Aug. 28-30, 1991, New Delhi, India. Submit two copies of extended abstract on rapid prototyping with functional programming languages or geometric pattern recognition by **Jan. 10, 1991,** and full paper by **Apr. 15, 1991,** to A.L. Lakshminarasimhan, AT&T Bell Labs, 480 Red Hill Rd., No. HR 2E030, Middletown, NJ 07748, phone (201) 615-4524.

**Compass 91, Sixth Conf. on Computer Assurance Systems Integrity, Software Safety, and Process Security:** June 25-27, 1991, Gaithersburg, Md. Cosponsors: IEEE Aerospace and Electronics Soc., IEEE Nat'l Capital Area Council. Submit five copies of paper by **Jan. 11, 1991,** to Roger U. Fujii, Logicon, 222 W. Sixth St., San Pedro, CA 90731, phone (213) 831-0611, ext. 2420, e-mail r.fujii@ieee.org.

**Sixth Int'l Conf. CAD/CAM, Robotics, and Factories of the Future:** Aug. 19-22, 1991, London. Sponsor: Int'l Soc. for Productivity Enhancement. Submit three copies of abstract by **Jan. 11, 1991,** and manuscript by **June 1, 1991,** to H. Bera, Mechanical Eng. Dept., South Bank Polytechnic, 103 Borough Rd., London SE1 0AA, UK, phone 011 (91) 81-928-8989, ext. 2095, fax 011 (91) 81-261-9115.

**Compsac 91, 15th Int'l Software and Applications Conf.:** Sept. 11-13, 1991, Tokyo. Cosponsor: Information Processing Soc. of Japan. Submit six copies of paper by **Jan. 12, 1991,** to Lionel M. Ni, Michigan State Univ., Computer Science Dept., A714 Wells Hall, East Lansing, MI 48824-1027, phone (517) 353-4386, fax (517) 336-1061, Internet ni@cps.msu.edu (for the Americas, Europe, and Africa); or Motoei Azuma, Waseda Univ., c/o Business Center for Academic Societies of Japan, 3-23-1 Hongo, Bunkyo-ku, Tokyo 113, Japan, phone 81 (3) 817-5831, fax 81 (3) 817-5836.

**ISS 91, Conf. on Information Sciences and Systems:** Mar. 20-22, 1991, Baltimore. Sponsor: Johns Hopkins Univ. Submit summary and regular or short paper by **Jan. 14, 1991,** to Frederick Davidson or John Goutsias, Electrical and Computer Eng. Dept., Johns Hopkins Univ., Baltimore, MD 21218, phone (301) 338-7871, fax (301) 338-5566.

**16th Int'l Symp. on Computer Systems:** Apr. 16-19, 1991, Monterrey, NL, Mexico. Sponsor: Inst. Tecnológico y de Estudios Superiores de Monterrey. Submit three copies of full paper by **Jan. 15, 1991,** to Carlos D. Hinojosa A., Dirección de Carrera ISC, ITESM, Suc. de Correos 'J', CP.64849, Monterrey, NL, Mexico, phone 52 (83) 58-2000, fax 52 (83) 58-8931.

**ICANN 91, Int'l Conf. on Artificial Neural Networks:** June 24-28, 1991, Espoo, Finland. Cosponsors: IEEE Neural Networks Council, Int'l Neural Network Soc. Submit manuscript by **Jan. 15, 1991,** to Olli Simula, Helsinki Univ. of Technology, SF-02150 Espoo, Finland, fax 358 (0) 451-3277, e-mail icann91@hutmc.hut.fi.

**RIDT 91, Second Int'l Workshop on Raster Imaging and Digital Typography:** Oct. 15-16, 1991, Boston. Cosponsor: Univ. of Massachusetts. Submittal deadline: **Jan. 15, 1991.** To obtain author information, contact Robert A. Morris, Math. and Computer Science Dept., Univ. of Massachusetts, Boston, MA 02125-3393, phone (617) 287-6466, e-mail ridt91-request@cs.umb.edu.

**ESEC 91, Third European Software Eng. Conf.:** Oct. 21-24, 1991, Milano, Italy. Sponsors: AFCET et al. Submit six copies of full paper and abstract by **Jan. 15, 1991,** to Alex van Lamsweerde, Unite d'Informatique, Univ. Catholique de Louvain, Place Sainte Barbe 2, B-1348 Louvain-La-Neuve, Belgium, e-mail esec@info.ucl.ac.be.

**Third Int'l Workshop on Artificial Intelligence in Real-Time Control:** Sept. 23-25, 1991, Sonoma Valley, Calif. Sponsor: Int'l Federation of Automatic Control. Submit four copies of extended abstract by **Jan. 15, 1991,** to Greg Suski, Lawrence Livermore Nat'l Lab, L-550, PO Box 808, Livermore, CA 94550, phone (415) 423-8070, e-mail suski@ocfmail.ocf.llnl.gov.

**Third Int'l Conf. on Software Eng. and Knowledge Eng.:** June 27-29, 1991, Skokie, Ill. Sponsors: Knowledge Systems Inst. et al. Submit abstract and four copies of complete paper by **Jan. 15, 1991,** to W.D. Hurley, Computer Science Dept., Alumni Hall, Univ. of Pittsburgh, Pittsburgh, PA 15260, phone (412) 624-8843, e-mail hurley@cs.pitt.edu.

**Int'l Conf. on the Performance of Distributed Systems and Integrated Comm. Networks:** Sept. 10-12, 1991, Kyoto, Japan. Sponsor: Int'l Federation for Information Processing. Submit five copies of full paper by **Jan. 15, 1991,** to Yutaka Takahashi, Applied Math and Physics Dept., Faculty of Eng., Kyoto Univ., Kyoto 606, Japan, phone 81 (75) 753-5493, fax 81 (75) 761-2437, e-mail yutaka@kuamp.kyoto-u.ac.jp.

**ICAIL 91, Third Int'l Conf. on Artificial Intelligence and Law:** June 25-28, 1991, Oxford, UK. Cosponsors: Soc. for Computer and Law (UK) et al. Submit five copies of paper by **Jan. 15, 1991,** to Marek Sergot, Computing Dept., Imperial College, 180 Queen's Gate, London, SW7 2BZ, UK, fax 44 (071) 581-8024, e-mail mjs@doc.ic.ac.uk.

**ICCIM 91, Int'l Conf. on Computer-Integrated Manufacturing:** Oct. 2-4, 1991, Singapore. Cosponsors: Gintic Inst. of CIM et al. Submit extended abstract by **Jan. 15, 1991,** to Lim Beng Siong, ICCIM 91, c/o Associated Conventions and Exhibitions Pte. Ltd., 204 Bukit Timah Rd., No. 04-00 Boon Liew Bldg., Singapore, phone (65) 732-6839, fax (65) 732-6309, e-mail bitnet%"gbslim@ntivax".

**1991 Int'l Conf. on Computer Processing of Chinese and Oriental Languages:** Aug. 13-16, 1991, Taipei, Taiwan. Cosponsors: Chinese Language Computer Soc. (USA) et al. Submit five copies of complete paper by **Jan. 15, 1991,** to Yaohan Chu, Computer Science Dept., Univ. of Maryland, College Park, MD 20742, phone (301) 405-2667, fax (301) 405-6707, e-mail ychu@cs.umd.edu.

**Second Int'l Information Research Conf.:** July 15-18, 1991, Cambridge, UK. Sponsors: British Library Research and Development Dept, Univ. of Pittsburgh. Submit detailed abstract by **Jan. 16, 1991,** to Karen Merry, British

Library R&D Dept., 2 Sheraton St., London W1V 4BH, UK, phone 44 (071) 323-7050, fax 44 (071) 323-7251.

**IAAI 91, Third Conf. on Innovative Applications of Artificial Intelligence:** July 15-17, 1991, Anaheim, Calif. Sponsor: Am. Assoc. for Artificial Intelligence. Submit five complete copies of paper by **Jan. 18, 1991,** to IAAI 91, AAAI, 445 Burgess Dr., Menlo Park, CA 94025-3496, phone (415) 328-3123.

**AI 91, Frontiers in Innovative Computing for the Nuclear Industry:** Sept. 15-18, 1991, Jackson, Wyo. Cosponsors: Am. Nuclear Soc. Idaho Section et al. Submit summary by **Jan. 18, 1991,** and full paper by **June 15, 1991,** to Richard W. Lindsay, Argonne Nat'l Lab, PO Box 2528, Idaho Falls, ID 83403-2528, phone (208) 526-7754, fax (208) 526-7623.

**29th Meeting of the Assoc. for Computational Linguistics:** June 18-21, 1991, Berkeley, Calif. Submit six copies of preliminary version of paper by **Jan. 19, 1991,** to Douglas E. Appelt, Artificial Intelligence Center, SRI Int'l, 333 Ravenswood Rd., Menlo Park, CA 94025, phone (415) 859-6150, fax (415) 859-6171, e-mail appelt@ai.sri.com.

**Sixth Int'l Workshop on Software Specification and Design:** Oct. 25-26, 1991, Como, Italy. Submit five copies of regular or position paper by **Jan. 21, 1991,** to Carlo Ghezzi, Dip. di Elettronica Politecnico di Milano, Piazza Leonardo Da Vinci 32, 20133 Milano, Italia, e-mail relett24@imipoli.bitnet.

**AAAI 91:** July 14-19, 1991, Anaheim, Calif. Sponsor: Am. Assoc. for Artificial Intelligence. Submit six complete copies of paper by **Jan. 30, 1991,** to AAAI 91, 445 Burgess Dr., Menlo Park, CA 94025-3496, phone (415) 328-3123.

**First Golden West Int'l Conf. on Intelligent Systems:** June 3-5, 1991, Reno, Nev. Sponsor: Int'l Soc. of Mini and Microcomputers. Submit three copies of preliminary version of paper by **Jan. 31, 1991,** to Carl G. Looney, CS Dept., Univ. of Nevada, Reno, NV 89557, e-mail looney@tahoe.unr.edu.

***The Visual Computer*** plans a special issue on visual user interface design tools. Submit six copies of article by **Jan. 31, 1991,** to Gurminder Singh, Inst. of Systems Science, Nat'l Univ. of Singapore, Kent Ridge, Singapore 0511, phone (65) 772-3651, e-mail issgs@nusvm.bitnet.

**Second Eurographics Workshop on Object-Oriented Graphics:** June 5-7, 1991, The Netherlands. Sponsor: Dutch Center for Math. and Computer Science (CWI). Submit four copies of full paper by **Jan. 31, 1991,** to Chris Laffra, Math. and Computer Science Dept., Leiden Univ., PO Box 9512, 2300RA Leiden, The Netherlands, fax 31 (71) 275-819, e-mail laffra@cs.leidenuniv.nl.

**22nd Pittsburgh Conf. on Modeling and Simulation:** May 2-3, 1991, Pittsburgh. Sponsor: Univ. of Pittsburgh. Submit two copies of abstract and summary by **Jan. 31, 1991,** to William G. Vogt or Marlin H. Mickle, Modeling and Simulation Conf., 348 Benedum Eng. Hall, Univ. of Pittsburgh, Pittsburgh, PA 15261.

***IEEE Software*** plans a special issue on software for performance analysis. Submit six copies of manuscript by **Feb. 1, 1991,** to Kathleen Nichols, Apple Computer, 20525 Mariani Ave., MS 76-3K, Cupertino, CA 95014, phone (408) 974-1136, e-mail nichols@apple.com; or Paul Oman, Computer Science Dept., College of Eng., Univ. of Idaho, Moscow, ID 83843, phone (208) 885-6589, e-mail oman@ted.cs.uidaho.edu.

***IEEE Trans. on Computers*** plans a special issue on artificial neural networks. Submit six copies of manuscript by **Feb. 1, 1991,** to Benjamin W. Wah, Coordinated Science Lab, MC228, Univ. of Illinois, 1101 W. Springfield Ave., Urbana, IL 61801-3082, phone (217) 333-3516, fax (217) 244-1764, e-mail wah%aquinas@uxc.cso.uiuc.edu.

**ICCD 91, IEEE Int'l Conf. Symp. on Computer Design:** Oct. 14-16, 1991, Cambridge, Mass. Cosponsors: IEEE Computer Soc. and IEEE Circuits and Systems Soc. Submit six copies of summary by **Feb. 1, 1991,** to Dwight Hill, AT&T Bell Labs, 3D-446, Murray Hill, NJ 07974, phone (201) 582-7766, e-mail dwight@research.att.com.

**ICGA 91, Fourth Int'l Conf. Symp. on Genetic Algorithms:** July 13-16, 1991, San Diego, Calif. Submit four copies of complete paper by **Feb. 1, 1991,** to Richard K. Belew, Computer Science and Eng. Dept., C-014, Univ. of California at San Diego, La Jolla, CA 92093, e-mail rik@cs.ucsd.edu.

**ITC 91, Int'l Test Conf.:** Oct. 28-Nov. 1, 1991, Nashville, Tenn. Cosponsor: IEEE Philadelphia Section. Submit paper by **Feb. 4, 1991,** to ITC 91, 1201 Sussex Turnpike, Suite 101, PO Box 264, Mount Freedom, NJ 07970, phone (201) 895-5260, fax (201) 895-7265.

**IEE Bicentennial Conf. on Computing:** July 1-3, 1991, London. Submit synopsis by **Feb. 4, 1991,** to Conf. Services, Institution of Electrical Engineers, Savoy Place, London WC2R 0BL, UK, phone 44 (71) 240-1871.

**Fifth Software Eng. Inst. Conf. on Software Eng.:** Oct. 7-8, 1991, Pittsburgh. Sponsor: SEI. Submit five copies of complete paper and abstract by **Feb. 4, 1991,** to James E. Tomayko, SEI, Carnegie Mellon Univ., Pittsburgh, PA 15213-3890, phone (412) 268-6806, fax (412) 268-5758, e-mail jet@sei.cmu.edu.

**ICAD 91, IFIP Working Conf. on Intelligent Computer-Aided Design:** Sept. 30-Oct. 3, 1991, Columbus, Ohio. Sponsors: Int'l Federation for Information Processing et al. Submit five copies of camera-ready paper by **Feb. 4, 1991,** to ICAD 91, Conferences and Institutes, Rm. 125, 1050 Carmack Rd., Columbus, OH 43210, phone (614) 929-1301, fax (614) 292-0492, e-mail sarah_sieling@gate.ce.ohio-state.edu.

**VLDB 91, 17th Int'l Conf. on Very Large Data Bases:** Sept. 3-6, 1991, Barcelona, Spain. Sponsor: IEEE Computer Soc. Tech. Committee on Data Eng. et al. Submit five copies of paper by **Feb. 15, 1991,** to Amilcar Semadas, INESC, Rua Alves Redol, 9, 7º, Apartado 10105, P-1017 Lisboa Codex, Portugal, e-mail inesc!acs%solo@relay.eu.net (US Internet), acs%solo@inesc.uucp (Europe Internet); or Guy M. Lohman, IBM Almaden Research Center, Dept. K55, Bldg. 801, 650 Harry Rd., San Jose, CA 95120-6099, e-mail lohman@ibm.com (Internet), lohman@almaden (Bitnet).

**Fifth Int'l Conf. on Fault-Tolerant Computing Systems:** Sept. 25-27, 1991, Nürnberg, Germany. Cosponsors: Gesellschaft für Informatik et al. Submit four copies of paper by **Feb. 15, 1991,** to M. Dal Cin, Univ. Erlangen — Nürnberg, IMMD III, Martensstr. 3, D-8520 Erlangen, Germany, fax 91 (31) 393-88, e-mail michel@uni-erlangen.de.

**Fourth Workshop on Computational Learning Theory:** Aug. 5-7, 1991, Santa Cruz, Calif. Submit 11 copies of abstract by **Feb. 15, 1991,** to L.G. Valiant, Aiken Computing Lab, Harvard Univ., Cambridge, MA 02138.

**1991 Electronic Packaging Conf.:** Sept. 15-19, 1991, San Diego, Calif. Sponsor: Int'l Electronics Packaging Soc. Submit eight copies of 300-word abstract by **Feb. 15, 1991,** to 1991 Program Committee, IEPS, 114 N. Hale St., Wheaton, IL 60187-5113, phone (708) 260-1044, fax (708) 260-0867.

**Sixth Conf. on Visual Comm. and Image Processing:** Nov. 10-13, 1991, Boston. Sponsors: Int'l Soc. for Optical Eng. et al. Submit four copies of 1,000-word extended summary by **Feb. 18, 1991,** to SPIE, PO Box 10, Bellingham, WA 98227-0010, phone (206) 676-3290, fax (206) 647-1445.

**OOPSLA 91, Sixth ACM Conf. on Object-Oriented Programming Systems, Languages, and Applications:** Oct. 6-11, 1991, Phoenix, Ariz. Submit six copies of full paper by **Mar. 1, 1991,** to Alan Snyder, Hewlett-Packard Labs, 1501 Page Mill Rd., PO Box 10490, Palo Alto, CA 94303-0969, phone (415) 857-8764, e-mail oopsla91@hplabs.hp.com.

**16th Conf. on Local Computer Networks:** Oct. 14-17, 1991, Minneapolis, Minn. Cosponsor: IEEE Computer Soc. Technical Committee on Computer Comm. Submit five copies of full paper by **Apr. 5, 1991,** to James F. Mollenauer, 16th LCN Conf., Artel Communications, 22 Kane Industrial Dr., Hudson, MA 01749, phone (508) 562-2100, fax (508) 562-6942.

**IEEE Infocom 92, 11th Conf. on Computer Comm.:** May 4-8, 1992, Florence, Italy. Cosponsor: IEEE Comm. Soc. Submit six copies of paper by **June 30, 1991,** to L. Fratta, Politecnico di Milano, c/o Cefriel, Via Emanueli, 15, 20126 Milano, Italy, phone 39 (2) 2399-3578, fax 39 (2) 2399-3587, e-mail fratta@imicefr.bitnet; or J. Kurose, Computer and Information Science Dept., Univ. of Massachusetts, Amherst, MA 01003, phone (413) 545-1585, e-mail kurose@cs.umass.edu.

# CALENDAR

In the accompanying Calendar and adjoining Call for Papers, the IEEE Computer Society logo identifies the conferences the society is participating in or sponsoring. Other conferences of interest to our readers, plus their sponsors, are also listed.

For inclusion in Call for Papers or Calendar, submit the following information: event name, date(s), location, and sponsor(s) as well as the phone and fax numbers and the electronic address of the person to contact. In addition, for Calls for Papers listings, include the name of the person to whom papers should be submitted and the deadline for submittals.

*Computer* should receive the above-mentioned information **at least five weeks before the month of publication** (i.e., for the **February 1991** issue, send information for receipt by **December 20, 1990**) to Chuck Governale, Calendar Dept., *Computer*, PO Box 3014, Los Alamitos, CA 90720-1264, fax (714) 821-4010, e-mail c.governale@compmail.com.

## December 1990

**1990 IEEE Workshop on Languages and Architectures for Automation, Dec. 19-21,** Honolulu. Sponsors: Pacific Int'l Center for High Technology Research et al. Contact D.Y.Y. Yun, Univ. of Hawaii, 711 Kapiolani Blvd., Suite 200, Honolulu, HI 96813-5249, phone (808) 539-1532, fax (808) 941-1399; or Shi-Kuo Chang, 322 Alumni Hall, Univ. of Pittsburgh, Pittsburgh, PA 15260, phone (412) 624-8493, fax (412) 624-8465, e-mail chang@vax.cs.pitt.edu.

**Seventh Israeli Conf. on Artificial Intelligence and Computer Vision, Dec. 26-27,** Tel Aviv, Israel. Contact A. Bruckstein, Faculty of Computer Science, Technion, 32000 Haifa, Israel, e-mail freddy@techsel.bitnet; or Shmuel Peleg, David Sarnoff Research Center, CN 5300, Princeton, NJ 08543-5300, phone (609) 734-2284, e-mail peleg@vision.sarnoff.com.

## January 1991

**Fourth CSI/IEEE Int'l Symp. on VLSI Design, Jan. 5-8,** New Delhi, India. Sponsors: Computer Soc. of India et al. Contact Yashwant K. Malaiya, Computer Science Dept., Colorado State Univ., Fort Collins, CO 80523, phone (303) 491-7031, fax (303) 491-2293, e-mail malaiya@ravi.cs.colostate.edu; or D. Roy Chowdhury, Gateway Design Automation, SDF#A-1, Noida Export Processing Zone, PO NEPZ, Noida 201305, India, phone 91 (05736) 62342, fax 91 (05736) 62343.

**SIAM Workshop on Automatic Differentiation of Algorithms, Jan. 7-9,** Breckenridge, Colo. Contact Soc. for Industrial and Applied Math., Conf. Coordinator, Dept. CC0590, 3600 University City Science Center, Philadelphia, PA 19104-2688, phone (215) 382-9800, fax (215) 386-7999, e-mail siamconfs@wharton.upenn.edu.

**Int'l Workshop on Formal Methods in VLSI Design, Jan. 9-11,** Puerto Rico. Cosponsors: ACM, IFIP. Contact P.A. Subrahmanyam, Rm. 4E-530, AT&T Bell Labs, Holmdel, NJ 07733, phone (201) 949-5812, fax (201) 949-3697, e-mail subra@vax135.att.com.

**Fifth Tech. Conf. on the X Window System, Jan. 14-16,** Boston. Sponsor: MIT X Consortium. Contact X Tech. Conf, Rm. 217, MIT Lab for Computer Science, 545 Technology Square, Cambridge, MA 02139.

**Int'l Conf. on Multimedia Information Systems, Jan. 16-18,** Singapore. Contact Desai Narasimhalu or Juzar Motiwalla, Inst. of Systems Science, Nat'l Univ. of Singapore, Heng Mui Keng Terr., Kent Ridge, Singapore 0511, phone (65) 772-2075, fax (65) 772-2002, Bitnet issad@nusvm.

**Int'l Workshop on Unix-Based Software Development Environments, Jan. 16-18,** Dallas. Sponsor: Usenix Assoc. Contact Usenix Conf. Office, 22672 Lambert St., Suite 613, El Toro, CA 92630, phone (714) 588-8649.

**Conf. on Optics, Electro-Optics, and Laser Applications in Science and Eng., Jan. 20-25,** Los Angeles. Sponsor: Int'l Soc. for Optical Eng. Contact SPIE, PO Box 10, Bellingham, WA 98227-0010, phone (206) 676-3290, fax (206) 647-1445.

**PADS, Workshop on Parallel and Distributed Simulation, Jan. 23-25,** Anaheim, Calif. Cosponsors: ACM, SCS. Contact Vijay Madisetti, School of Electrical Eng., Georgia Inst. of Tech., Atlanta, GA 30332-0250, phone (404) 853-9830, fax (404) 894-8363, e-mail vijaykm@petri.gatech.edu; or David M. Nicol, Computer Science Dept., College of William and Mary, Williamsburg, VA 23185, phone (804) 221-3458, e-mail nicol@cs.wm.edu.

**Winter 91 Unix Tech. Conf., Jan. 21-25,** Dallas. Sponsor: Usenix Assoc. Contact Usenix Conf. Office, 22672 Lambert St., Suite 613, El Toro, CA 92630, phone (714) 588-8649.

**Second ACM-SIAM Symp. on Discrete Algorithms, Jan. 28-30,** San Francisco. Contact SIAM Conf. Coordinator, Dept. CC0590, 3600 University City Science Center, Philadelphia, PA 19104-2688, phone (215) 382-9800, fax (215) 386-7999, e-mail siamconfs@wharton.upenn.edu.

**IEEE Int'l Conf. on Wafer Scale Integration, Jan. 29-31,** San Francisco. Cosponsors: IEEE Components, Hybrids, and Manufacturing Technology Soc. Contact Terry Chappell, 730 Encino Dr., Aptos, CA 95003, phone (408) 662-1936; or R. Mike Lea, Brunel Univ., Uxbridge UB8 3PH, UK, phone (44) 895-74000, ext. 2821, fax (44) 895-58728, e-mail mike.lea@brunel.ac.uk.

**Nat'l Debate on Achieving Quality Software, Jan. 29-Feb. 1,** San Diego, Calif. Cosponsors: Soc. for Software Quality, Am. Soc. for Quality Control. Contact Martin Einhorn, 7373 University Ave., Suite 213, La Mesa, CA 92041, phone (619) 697-0085.

## February 1991

**WCF 91, Western Comm. Forum, Feb. 4-6,** Phoenix, Ariz. Sponsor: Nat'l Eng. Consortium. Contact NEC, 303 E. Wacker Dr., Suite 740, Chicago, IL 60601, phone (312) 938-3500, fax (312) 938-8787.

**Systems/USA Tech. Conf., Feb. 11-13,** Anaheim, Calif. Sponsor: Am. Electronics Assoc. Contact AEA, 5201 Great America Pkwy., Santa Clara, CA 95054, phone (503) 359-5873 or (408) 987-4204, fax (503) 357-3839 or (408) 970-8565.

**Fifth Int'l Conf. on Modeling Techniques and Tools for Computer Performance Evaluation, Feb. 13-15,** Torino, Italy. Contact Maria Carla Calzarossa, Dip. di Informatica e Sistemistica, Univ. di Pavia, Via Abbiategrasso, 209, 27100 Pavia, Italy, phone 39 (382) 391-350, fax 39 (382) 422-881, e-mail mcc@ipvpel.infn.it.

**IWPT 91, Second Int'l Workshop on Parsing Technologies, Feb. 13-15,** Cancun, Mexico. Contact Joan Maddamma, IWPT 91, School of Computer Science, Carnegie Mellon Univ., Pittsburgh, PA 15213, phone (412) 268-7656, fax (412) 621-5473, e-mail jfm@cs.cmu.edu.

**ISSCC 91, 1991 IEEE Int'l Solid-State Circuits Conf., Feb. 13-15,** San Francisco. Sponsors: IEEE Solid-State Circuits Council et al. Contact Diane Suiters, Courtesy Associates, 655 15th St. NW, Suite 300, Washington, DC 20005, phone (202) 639-4255.

**PCCS 1, First Int'l Workshop on Performability Modeling of Computer and Comm. Systems, Feb. 18-19,** Enschede, The Netherlands. Contact Nico M. van Dijk, Free Univ., Faculty of Economics, PO Box 7161, 1007 MC Amsterdam, The Netherlands, phone 31 (20) 548-7061, fax 31 (20) 462-645, e-mail ectricvu@sara.nl.

**CAIA 91, Seventh IEEE Conf. on Artificial Intelligence Applications, Feb. 24-28,** Miami Beach, Fla. Contact IEEE Computer Soc., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

**Fourth Topical Meeting on Robotics and Remote Systems for Hazardous Environments, Feb. 24-28,** Albuquerque, N.M. Contact Raymond W. Harrigan, Div. 1414, Sandia Nat'l Labs, Albuquerque, NM 87185, phone (505) 846-6278, fax (505) 846-7425.

**EDAC 91, European Design Automation Conf., Feb. 25-28,** Amsterdam. Sponsor: Institution of Electrical Engineers. Contact Secretariat, EDAC 91, CEP Consultants, 26-28 Albany St., Edinburgh EH1 3QH, Scotland, fax 44 (31) 557-5749.

**Compcon Spring 91, Feb. 25-Mar. 1,** San Francisco. Contact Compcon Spring 91, IEEE Computer Soc., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

## March 1991

**First Great Lakes Symp. on VLSI, Mar. 1-2,** Kalamazoo, Mich. Contact Eltayeb S. Abuelyaman, Electrical Eng. Dept., Eastern Michigan Univ., Kalamazoo, MI 49007, fax (616) 387-4024.

**Fifth Int'l Workshop on High-Level Synthesis, Mar. 3-6,** Buhlerhohe, Germany. Cosponsors: IEEE et al. Contact Raul Camposano, IBM T.J. Watson Research Center, PO Box 218, Yorktown Heights, NY 10598, phone (914) 945-3871, e-mail raulc@ibm.com.

**22nd Tech. Symp. on Computer Science Education, Mar. 7-8,** San Antonio, Texas. Sponsor: ACM SIGSCE. Contact Nell Dale, Computer Science Dept., Univ. of Texas at Austin, Austin, TX 78712, phone (512) 471-9539, e-mail ndale@cs.utexas.edu.

**Fourth Computer Virus and Security Conf., Mar. 14-15,** New York City. Sponsor: Data Processing Management Assoc. Financial Industries. Contact Judy S. Brand, PO Box 6313, FDR Station, New York, NY 10150, phone (800) 835-2246.

**Third IEE Conf. on Telecomm., Mar. 17-20,** Edinburgh, Scotland. Sponsor: Institution of Electrical Engineers. Contact Conf. Services, IEE, Savoy Place, London WC2R 0BL, UK, phone 44 (71) 240-1871, fax 44 (71) 240-7735.

**Symp. on Experiences with Distributed and Multiprocessor Systems, Mar. 21-22,** Atlanta. Sponsor: Usenix Assoc. Contact George Leach, AT&T Paradyne, MS LG-129, PO Box 2826, Largo, FL 34649-2826, phone (813) 530-2376, e-mail reggie@pdn.paradyne.com.

**Fifth SIAM Conf. on Parallel Processing and Scientific Computing, Mar. 25-27,** Houston. Contact Soc. for Industrial and Applied Math. Conf. Coordinator, Dept. CC0590, 3600 University City Science Center, Philadelphia, PA 19104-2688, phone (215) 382-9800, fax (215) 386-7999, e-mail siamconfs@wharton.upenn.edu.

**Advanced Research in VLSI Conf., Mar. 25-27,** Santa Cruz, Calif. Sponsors: Univ. of California at Santa Cruz, Univ. of California at Berkeley. Contact Kevin Karplus, Computer Eng., Univ. of California at Santa Cruz, Santa Cruz, CA 95064, Internet karplus@ce.ucsc.edu.

**Auto Carto 10, 10th Int'l Symp. on Automated Cartography, Mar. 25-28,** Baltimore. Cosponsors: Am. Cartographic Assoc. et al. Contact ACSM/ASPRS/Auto Carto 10, 5410 Grovesnor Lane, Bethesda, MD 20814.

**CEEDA 91, Int'l Conf. on Concurrent Eng. and Electronic Design Automation, Mar. 26-28,** Bournemouth, Dorset, UK. Sponsors: Institution of Electrical Engineers et al. Contact Sa'ad Medhat, Bournemouth Polytechnic, Poole House, Talbot Campus, Fern Barrow, Dorset BH12 5BB, UK, phone 44 (81) 595-492, fax 44 (81) 595-368, e-mail saiad medhat@eurolom.ie.

**10th IEEE Int'l Phoenix Conf. on Computers and Comm., Mar. 27-30,** Scottsdale, Ariz. Sponsors: IEEE, IEEE Comm. Soc. Contact Oris Friesen, Bull HN, PO Box 8000, MS A93, Phoenix, AZ 85066, phone (602) 862-5200, e-mail friesen@system-m.phx.bull.com.

## April 1991

**24th Computer Simulation Conf., Apr. 1-5,** New Orleans. Sponsor: Soc. for Computer Simulation. Contact George W. Zobrist, Computer Science Dept., Univ. of Missouri at Rolla, Rolla, MO 65401, phone (314) 341-4836, e-mail c2816@umrvmb.umr.edu.

**Dasfaa 91, Second Int'l Symp. on Database Systems for Advanced Applications, Apr. 2-4,** Tokyo. Sponsor: Information Processing Soc. of Japan. Contact Yahiko Kambayashi, Computer Science Dept., Kyushu Univ., 6-10-1 Hakozaki, Higashi Fukuoka 812, Japan, phone 81 (92) 641-1101, ext. 5407; or Yoshifumi Masunaga, Univ. of Library and Information Science, 1-2 Kasuga, Tsukuba, Ibaraki 305, Japan, phone 81 (298) 52-0511, ext. 340, fax 81 (298) 52-4326, e-mail masunaga@ulis.ac.jp.

**Flairs 91, Florida Artificial Intelligence Research Symp., Apr. 2-5,** Cocoa Beach, Fla. Sponsor: Florida Artificial Intelligence Research Soc. Contact Avelino J. Gonzalez, Computer Eng. Dept., Univ. of Central Florida, Orlando, FL 32816, phone (407) 281-5027, e-mail fdgonzal%ucf1vm.bitnet@cunyvm.cuny.edu.

**SAC 91, 1991 Symp. on Applied Computing, Apr. 3-5,** Kansas City, Mo. Sponsor: Univ. of Missouri — Kansas City. Contact Richard G. Hetherington, SAC 91, Univ. of Missouri — Kansas City, Computer Science Telecommunications Program, 5100 Rockhill Rd., Kansas City, MO 64110-2499, phone (816) 235-2399.

**Third Symp. on Integrated Ferroelectrics, Apr. 3-5,** Colorado Springs, Colo. Contact Conf. Secretary, Microelectronics Research Lab, Univ. of Colorado at Colorado Springs, PO Box 7150, Colorado Springs, CO 80933-7150, phone (719) 593-3488, fax (719) 594-4257.

**Computer Graphics and Education 91, Apr. 4-6,** Barcelona, Spain. Sponsor: Int'l Federation for Information Processing. Contact Steve Cunningham, Computer Science Dept, California State Univ. at Stanislaus, Turlock, CA 95380, phone (209) 667-3176, e-mail rsc@altair.csustan.edu; or Roger Hubbold, Computer Science Dept., Univ. of Manchester, Oxford Road, Manchester M13 9PL, UK, phone (44) 61-275-6158, e-mail hubbold@uk.ac.man.cs.

**IEEE Infocom 91, Conf. on Computer Comm., Apr. 7-11,** Miami, Fla. Cosponsors: IEEE Computer Soc. and Comm. Soc. Contact N. Shacham, IEEE Infocom 91, SRI Int'l, 333 Ravenswood Ave., Menlo Park, CA 94025, phone (415) 859-5710, e-mail shacham@sri.com.

**1991 IEEE Int'l Conf. on Robotics and Automation, Apr. 7-12,** Sacramento, Calif. Sponsor: IEEE Robotics and Automation Soc. Contact Robotics and Automation, PO Box 3216, Silver Spring, MD 20918, phone (301) 434-1990.

**IMS 91, First IEEE Int'l Workshop on Interoperability in Multidatabase Systems, Apr. 8-9,** Kyoto, Japan. Contact Ahmed K. Elmagarmid, Purdue Univ., Computer Sciences Dept., West Lafayette, IN 47907, phone (317) 494-1998; or Yutaka Matsushita, Instrumentation Dept., Keio Univ., Hiyoshi, Yokohama, Japan, phone 81 (44) 63-1141, ext. 3564.

**DCC 91, Data Compression Conf., Apr. 8-10,** Snowbird, Utah. Sponsor: IEEE Computer Soc. Tech. Committee on Computer Comm., NASA/CESDIS. Contact Martin Cohn, Computer Science Dept., Brandeis Univ., Waltham, MA 02254, phone (617) 736-2705, fax (617) 736-2741, e-mail marty@cs.brandeis.edu.

**ASPLOS 4, Fourth Int'l Conf. on Architectural Support for Programming Languages and Operating Systems, Apr. 8-11,** Santa Clara, Calif. Sponsor: ACM.

Contact Bob Rau, Hewlett-Packard Labs, 1501 Page Mill Rd., Bldg. 3U, Palo Alto, CA 94304, fax (415) 857-8558, e-mail rau@hplabs.hp.com.

**Seventh Int'l Conf. on Data Eng., Apr. 8-12,** Kobe, Japan. Contact Ming T. (Mike) Liu, Computer and Information Science Dept., Ohio State Univ., 2036 Neil Ave., Columbus, OH 43210-1277, phone (614) 292-1837, e-mail liu@cis.ircc.ohio-state.edu; or Data Eng. 91, IEEE Computer Soc., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013, fax (202) 728-0884.

**IFIP Working Conf. on Modeling in Computer Graphics, Apr. 8-12,** Tokyo. Sponsor: IFIP TC 5/WG 5.10. Contact Tosiyasu L. Kunii, Information Science Dept., Faculty of Tokyo, Univ. of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan, phone 81 (3) 816-1783, fax 81 (3) 818-4607, e-mail b39756@tansei.cc.u-tokyo.ac.jp.

**ETC 91, 1991 European Test Conf., Apr. 10-12,** Munich, Germany. Sponsor: VDE (Zentralstelle Tagungen und Seminare). Contact Peter Stilke, VDE, Stresemannallee 15, D-6000 Frankfurt 70, Germany, phone (69) 6308-203, fax (69) 6308-273.

**RTA 91, Fourth Int'l Conf. on Rewriting Techniques and Applications, Apr. 10-12,** Como, Italy. Sponsor: State Univ. of Milan. Contact G. Degli Antoni or Marelva Bianchi, Dip. di Scienze Dell' Informazione, Univ. di Milano, Via Milano Moretto da Brescia 9, I-20133 Milano, Italia, phone 39 (02) 7575-201, fax 39 (02) 7611-0556, e-mail gdantoni@imisiam.bitnet.

**14th IEEE Workshop on Design for Testability, Apr. 15-18,** Vail, Colo. Contact T.W. Williams, IBM, PO Box 1900, Dept. J22/02SR, Boulder, CO 80301-9191.

**Ninth IEEE VLSI Test Symp., Apr. 16-18,** Atlantic City, N.J. Cosponsor: IEEE Philadelphia Section. Contact Mukund Modi, Naval Air Eng. Center, ATE Software Center, Code: 52514, Lakehurst, NJ 08733, phone (201) 323-7002, fax (201) 323-7445.

**16th Int'l Symp. on Computer Systems, Apr. 16-19,** Monterrey, NL, Mexico. Sponsor: Inst. Tecnológico y de Estudios Superiores de Monterrey. Contact Carlos D. Hinojosa A., Dirección de Carrera ISC, ITESM, Suc. de Correos 'J', CP.64849, Monterrey, NL, Mexico, phone 52 (83) 58-2000, fax 52 (83) 58-8931.

**CHDL 91, 10th Int'l Symp. on Computer Hardware Description Languages and their Applications, Apr. 22-24,** Marseille, France. Cosponsors: Int'l Federation for Information Processing et al. Contact Dominique Borrione, Imag/Artemis, BP 53X, 38041 Grenoble Cedex, France, phone (33) 7651-4604, ext. 5240, fax (33) 7651-9637, e-mail borrione@imag.imag.fr.

**Second European Distributed Memory Computing Conf., Apr. 22-24,** Munich, Germany. Cosponsors: Gesellschaft fur Informatik et al. Contact Arndt Bode, Computer Science, Technische Univ. Munich, POB 20-24-20, D-8000 Munich 2, Germany, e-mail bode@infovax.informatik.tumuenchen.dbp.de.

**KMET 91, Int'l Conf. on Knowledge Modeling and Expertise Transfer, Apr. 22-24,** Sophia Antipolis, France. Cosponsors: Assoc. Francaise pour la Cybernetique Economique et Technique et al. Contact KMET 91, Univ. de Nice, Sophia Antipolis, CNRS, 13S-LISAN, Daniele Herin-Aime, bat. 4, rue A. Einstein, 06560, Valbonne, France, fax (33) 92-94-28-98, e-mail dh@cerisi.cerisi.fr.

**Second Int'l Conf. on Systems Integration, Apr. 22-25,** Morristown, N.J. Cosponsors: New Jersey Inst. of Technology et al. Contact Peter A. Ng, Computer and Information Science Dept., New Jersey Inst. of Technology, University Heights, Newark, NJ 07102, phone (201) 596-3387, e-mail ng_p@vienna.njit.edu.

**NCGA 91, 1991 Nat'l Computer Graphics Assoc. Conf., Apr. 22-25,** Chicago. Contact NCGA, 2722 Merrilee Dr., Suite 200, Fairfax, VA 22031, phone (800) 225-6242 or (703) 698-9600.

**CHI 91, 1991 Conf. on Human Factors in Computing Systems, Apr. 27-May 2,** New Orleans. Sponsor: ACM. Contact Keith Butler, Boeing, Advanced Technology Center, PO Box 24346, M/S 7L-64, Seattle, WA 98124, phone (206) 865-3389; or June Davis, 13 Annapolis St., Annapolis, MD 21401, phone (301) 269-6801.

**ECF 91, Eastern Comm. Forum, Apr. 29-May 1,** Washington, DC. Sponsor: Nat'l Eng. Consortium. Contact NEC, 303 E. Wacker Dr., Suite 740, Chicago, IL 60601, phone (312) 938-3500, fax (312) 938-8787.

**Int'l Conf. on Cognitive Sciences, Apr. 29-May 2,** Montreal. Cosponsors: AFCET et al. Contact Gilles Gauthier, Math. and Computer Science Dept., Univ. of Quebec, PO Box 8888, Station A, Montreal, Que., Canada H3C 3P8, phone (514) 987-8212, fax (514) 987-8477.

**Fifth Int'l Parallel Processing Symp., Apr. 30-May 2,** Anaheim, Calif. Sponsor: IEEE Computer Soc. Orange County Chapter. Contact Larry H. Canter, Computer Systems Approach, 1140 S. Raymond Ave., Suite B, Fullerton, CA 92631, phone (714) 738-3414, fax (714) 738-3470.

## May 1991

**22nd Pittsburgh Conf. on Modeling and Simulation, May 2-3,** Pittsburgh. Sponsor: Univ. of Pittsburgh. Contact William G. Vogt or Marlin H. Mickle, Modeling and Simulation Conf., 348 Benedum Eng. Hall, Univ. of Pittsburgh, Pittsburgh, PA 15261.

**Fifth SIAM Int'l Symp. on Domain Decomposition Methods for Partial Differential Equations, May 6-8,** Norfolk, Va. Contact Soc. for Industrial and Applied Math., Dept. CC0590, 3600 University City Science Center, Philadelphia, PA 19104-2688, phone (215) 382-9800, fax (215) 386-7999, e-mail siamconfs@wharton.upenn.edu.

**SID 91, 1991 Int'l Symp., Seminar, and Exhibition, May 6-10,** Anaheim, Calif. Sponsor: Soc. for Information Display. Contact SID, c/o Palisades Inst. for Research Services, 201 Varick St., Suite 1140, New York, NY 10014, phone (212) 620-3371, fax (212) 620-3379.

**IEEE Pacific Rim Conf. on Comm., Computers, and Signal Processing, May 9-10,** Victoria, B.C., Canada. Cosponsors: IEEE Victoria Section, Univ. of Victoria. Contact Technical Program Committee, c/o Pan Agathoklis, Electrical and Computer Eng. Dept., Univ. of Victoria, PO Box 3055, Victoria, B.C., Canada V8W 3P6, phone (604) 721-8618, fax (604) 721-8676.

**CBMS 91, Fourth IEEE Symp. on Computer-Based Medical Systems, May 12-14,** Baltimore. Cosponsors: IEEE Computer Soc., IEEE Eng. in Medicine and Biology Soc., and IEEE Baltimore Section. Contact Jeffery C. Lesho, Johns Hopkins Univ., Applied Physics Lab., Bldg. 2-257, Johns Hopkins Rd., Laurel, MD 20723-6099, phone (301) 953-5000, ext. 8057.

**CICC 91, IEEE Custom Integrated Circuits Conf., May 12-15,** San Diego, Calif. Contact Jim Lipman, VLSI Technology, 1109 McKay Dr. MS-32, San Jose, CA 95131, phone (408) 434-7673.

**44th Conf. of the Soc. for Imaging Science and Technology, May 12-17,** St. Paul, Minn. Contact SPSE, 7003 Kilworth Lane, Springfield, VA 22151, phone (703) 642-9090, fax (703) 642-9094.

**ICSE 13, 13th Int'l Conf. on Software Eng., May 13-17,** Austin, Texas. Cosponsor: ACM. Contact ICSE 13, Bryan Fugate, MCC, 3500 W. Balcones Center Dr., Austin, TX 78759-6509, phone (512) 338-3330; MCC, PO Box 200015, Austin, TX 78720-0015; or ICSE 13, IEEE Computer Soc., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

**CompEuro 91, IEEE Int'l Conf. on Advanced Computer Technology, Reliable Systems, and Applications, May 13-17,** Bologna, Italy. Cosponsors: IEEE Region 8 et al. Contact CompEuro 91 Conf. Secretariat, c/o Sercoop, via Crociali 2, 40138 Bologna, Italy, phone 39 (51) 300-811, fax 39 (51) 309-477; or Vito A. Monaco, Dip. di Elettronica Informatica E Sistemistica, Univ. di Bologna, Viale Risorgimento 2, 40136, Bologna, Italy, fax 39 (51) 644-3073.

**Ada-Europe Athens 91 Conf., May 13-17,** Athens. Cosponsors: Ada-Europe et al. Contact Z. Kaplanidis, Zita Tourist Club, 46 Voulis St., GR - 10558 Athens, Greece, phone 30 (1) 323-9744/7, fax 30 (1) 324-1720.

**North Am. Fuzzy Information Processing Soc. Workshop, May 15-17,** Columbia, Mo. Contact Jim Keller, Electrical and Computer

Eng., Univ. of Missouri — Columbia, Columbia, MO 65211, phone (314) 882-7339, fax (314) 882-0397.

**Int'l Symp. on Software Reliability Eng., May 17-18,** Austin, Texas. Cosponsors: IEEE Computer Soc. Tech. Committee on Software Eng. and the Nat'l Aeronautics and Space Administration. Contact Anneliese von Mayrhauser, Computer Science Dept., Illinois Inst. of Technology, 600 S. Lambert Rd., Glen Elyn, IL 60137, phone (708) 790-4385, e-mail csavm@karl.iit.edu.

**Workshop on Parallel and Distributed Debugging, May 20-21,** Santa Cruz, Calif. Cosponsors: ACM, US Navy Office of Naval Research. Contact Bart Miller, Computer Science Dept., Univ. of Wisconsin, 1210 W. Dayton St., Madison, WI 53706, phone (608) 263-3378, Internet bart@cs.wisc.edu.

**1991 IEEE Symp. on Research in Security and Privacy, May 20-22,** Oakland, Calif. Sponsor: IEEE Computer Soc. Tech. Committee on Security and Privacy. Contact Daniel Schnackenberg, Boeing, MS 9P-64, PO Box 3999, Seattle, WA 98124, phone (206) 657-5595, e-mail schnackenberg@dockmaster.ncsc.mil.

**Second Physical Design Workshop, May 20-22,** Laurel Highlands, Pa. Sponsor: ACM SIGDA. Contact Antun Domic, HLO2-3J3, DEC, 77 Reed Rd., Hudson, MA 01749, e-mail domic@cadsys.dec.com.

**ICDCS 91, 11th Int'l Conf. on Distributed Computing Systems, May 20-24,** Arlington, Texas. Contact Bill D. Carroll, Computer Science Eng. Dept., Univ. of Texas at Arlington, Box 19015, Arlington, TX 76019-0015, phone (817) 273-3785, e-mail carroll@evax.ari.utexas.edu.

**SESAW, Fourth Software Eng. Standard Application Workshop, May 21-23,** San Diego, Calif. Contact Vera V. Edelstein, Nynex, 500 Westchester Ave., White Plains, NY 10604, phone (914) 683-2888.

**SCAI 91, Third Scandinavian Conf. on Artificial Intelligence, May 21-24,** Roskilde, Denmark. Contact Brian Mayoh, Computer Science Dept., Aarhus Univ., Ny Munkegade, Bldg. 540, DK-8000 Aarhus C, Denmark, phone 45 (86) 127188, fax 45 (86) 135725, e-mail brian@daimi.aau.dk.

**Second Int'l Conf. on Algebraic Methodology and Software Technology, May 22-24,** Iowa City, Iowa. Contact Teodor Rus, Computer Science Dept., Univ. of Iowa, Iowa City, IA 52242, phone (319) 335-0694, e-mail rus@herky.cs.uiowa.edu.

**Melecon 91, Fifth Mediterranean Electrotechnical Conf., May 22-24,** Ljubljana, Yugoslavia. Cosponsors: IEEE Region 8 Yugoslavia Section. et al. Contact Melecon 91 Secretariat, Fakulteta za elektrotehniko, Trzaska 25, 61001 Ljubljana, Yugoslavia, fax 38 (61) 264-990.

**Computer Animation 91, May 22-25,** Geneva, Switzerland. Cosponsors: Computer Graphics Soc. Contact Nadia M. Thalmann, Mira Lab CUI, Univ. of Geneva, 12 rue du Lac, CH 1207, Geneva, Switzerland, phone 41 (22) 787-6581, fax 41 (22) 735-3905, e-mail thalmann@uni2a.unige.ch.

**21st Int'l Symp. on Multiple-Valued Logic, May 26-29,** Victoria, Canada. Contact D.M. Miller, Computer Science Dept., Univ. of Victoria, PO Box 1700, Victoria, B.C., Canada, V8W 2Y2, phone (604) 721-7220, fax (604) 721-7292, e-mail dmill@csr.uvic.cdn.

**ISCA 18, 18th Int'l Symp. on Computer Architecture, May 26-30,** Toronto. Cosponsor: ACM. Contact K.C. Smith, Univ. of Toronto, Electrical Eng. Dept., Toronto, Ont. M5S 1A4, Canada, phone (416) 978-5033.

**ICCI 91, Int'l Conf. on Computing and Information, May 27-29,** Ottawa, Canada. Sponsors: Carleton Univ, Ottawa; Natural Sciences and Eng. Research Council of Canada. Contact Frank Fiala, School of Computer Science, Carleton Univ., Ottawa, Canada K1S 5B6, phone (613) 788-4333, fax (613) 788-4334, e-mail icci@scs.carleton.ca.

**Fifth Israel Conf. on Computer Systems and Software Eng., May 28-29,** Herzlia, Israel. Sponsors: IEEE Computer Soc. Israeli Chapter et al. Contact M. Winokur, c/o ORTRA, PO Box 50432, Tel Aviv 61500, Israel, phone 972 (3) 664-825, fax 972 (3) 660-952.

## June 1991

**Workshop on Directions in Automated CAD-Based Vision, June 2-3,** Maui, Hawaii. Contact Linda Shapiro, Computer Science and Eng. Dept., FR-35, Univ. of Washington, Seattle, WA 98195, phone (206) 543-2196, fax (206) 543-3842.

**Fourth Int'l Conf. on Industrial and Eng. Applications of Artificial Intelligence and Expert Systems, June 2-5,** Kauai, Hawaii. Sponsors: ACM et al. Contact Moonis Ali, Univ. of Tennessee Space Inst., MS15, B.H. Goethert Pkwy., Tullahoma, TN 37388-8897, phone (615) 455-0631, ext. 236, fax (615) 454-2354, e-mail alif@utsiv1.bitnet.

**11th Int'l Conf. on Decision-Support Systems, June 3-5,** Manhattan Beach, Calif. Sponsor: Inst. for Management Sciences. Contact TIMS, 290 Westminster St., Providence, RI 02903.

**First Golden West Int'l Conf. on Intelligent Systems, June 3-5,** Reno, Nev. Sponsor: Int'l Soc. of Mini and Microcomputers. Contact Carl G. Looney, CS Dept., Univ. of Nevada, Reno, NV 89557, phone (702) 784-6927, e-mail looney@tahoe.unr.edu.

**CVPR 91, IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition, June 3-7,** Lahaina, Maui, Hawaii. Contact Shahriar Negahdaripour, Electrical Eng. Dept., Univ. of Hawaii at Manoa, 2540 Dole St., Honolulu, HI 96822, e-mail shahriar@wiliki.eng.hawaii.edu.

**20th Mumps Users Group Meeting, June 3-7,** New Orleans. Contact Mumps Users Group, 4321 Hartwick Rd., Suite 100, College Park, MD 20740, phone (301) 779-6555, fax (301) 779-7674.

**Symp. on Solid Modeling Foundations and CAD/CAM Applications, June 5-7,** Austin, Texas. Sponsor: ACM SIGGraph. Contact Joshua Turner, CII 7015, RDRC, Rensselaer Polytechnic Inst., Troy, NY 12180-3590, phone (518) 276-6751, fax (518) 276-2702, e-mail jturner@rdrc.rpi.edu.

**Second Eurographics Workshop on Object-Oriented Graphics, June 5-7,** The Netherlands. Sponsor: Dutch Center for Math. and Computer Science (CWI). Contact Marja Hegt, CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands, phone 31 (20) 592-4058, fax 31 (20) 592-4199, e-mail marja@cwi.nl.

**Parle 91, Conf. on Parallel Architectures and Languages Europe, June 10-13,** Eindhoven, The Netherlands. Cosponsors: Commission of European Communities et al. Contact F. Stoots, Philips Research Labs, PO Box 80.000, 5600 JA Eindhoven, The Netherlands, fax 31 (40) 744-758, e-mail stoots@dooma.prl.philips.nl.

**ISCAS 91, 24th IEEE Int'l Symp. on Circuits and Systems, June 11-14,** Singapore. Sponsor: IEEE Circuits and Systems Soc. Contact ISCAS 91 Secretariat, Comm. Int'l Associates, 44/46 Tanjong Pagar Rd., Singapore 0208, phone (65) 226-2823, fax (65) 226-2877.

**SCM 3, Third Int'l Software Configuration Management Workshop, June 12-14,** Trondheim, Norway. Cosponsors: ACM, et al. Contact Reidar Conradi, Computer Systems and Telematics Div., Norwegian Inst. of Technology, N-7034 Trondheim, Norway, phone 47 (7) 593-444; or Peter Feiler, Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA 15213-3890, phone (412) 268-7790, e-mail phf@sei.cmu.edu.

**1991 ACM Symp. on Personal and Small Computers, June 12-14,** Toronto. Cosponsors: Nat'l Research Council of Canada et al. Contact Michael Bauer, Computer Science Dept., Middlesex College, Univ. of Western Ontario, London, Ont., Canada N6A 5B7, e-mail bauer@csd.uwo.ca or bauer@uwovax.bitnet.

**DAC 91, 28th ACM/IEEE Design Automation Conf., June 16-21,** San Francisco. Contact MP Associates, 7490 Clubhouse Rd., Suite 102, Boulder, CO 80301, phone (303) 530-4333.

**1991 ACM Int'l Conf. on Supercomputing, June 17-21,** Cologne, Germany. Cosponsors: Gesellschaft für Informatik et al. Contact Ruediger Esser, FKA-ZAM, D-5170 Juelich, Germany, phone 49 (2461) 61-6588, fax 49 (2461) 61-6656, e-mail zdv003@djukfa11.bitnet.

**Eighth Int'l Conf. on Testing Computer Software, June 17-21,** Washington, DC. Sponsor: Data Processing Management Assoc. Educational Foundation. Contact Genevieve Houston-Ludlam, Frontier Technologies, 190 Admiral Cochran Dr., Suite 180, Annapolis, MD 21401, phone (301) 266-8244, fax (301) 224-3840.

**29th Meeting of the Assoc. for Computational Linguistics, June 18-21,** Berkeley, Calif. Contact Don Walker, Bellcore, MRE 2A379, 445 South St., Box 1910, Morristown, NJ 07960-1910, phone (201) 829-4312, e-mail walker@flash.bellcore.com.

**CGI 91, Int'l Conf. on Computer Graphics, June 22-28,** Cambridge, Mass. Cosponsors: Computer Graphics Soc., MIT. Contact Barbara Dullea, Ocean Eng. Dept., MIT Rm. 5-435, 77 Massachusetts Ave., Cambridge, MA 02139, fax (617) 253-8125, e-mail barbara@deslab.mit.edu.

**1991 IEEE Int'l Symp. on Information Theory, June 23-28,** Budapest, Hungary. Contact Anthony Ephremides, Electrical Eng. Dept., Univ. of Maryland, College Park, MD 20742, phone (301) 405-3641, arpanet tony@eng.umd.edu.

**ICANN 91, Int'l Conf. on Artificial Neural Networks, June 24-28,** Espoo, Finland. Cosponsors: IEEE Neural Networks Council, Int'l Neural Network Soc. Contact Congress Management Systems, PO Box 151, SF-00141 Helsinki, Finland, fax 358 (0) 170-122.

**FTCS 21, 21st Int'l Symp. on Fault-Tolerant Computing, June 25-27,** Montreal. Sponsor: IEEE Computer Soc. Tech. Committee on Fault-Tolerant Computing. Contact Vinod K. Agarwal, McGill Univ., Electrical Eng. Dept., 3480 University St., Montreal, Que., Canada H3A 2A7, phone (514) 398-7136, fax (514) 398-4470, e-mail agarwal@spock.ee.mcgill.ca.

**Compass 91, Sixth Conf. on Computer Assurance Systems Integrity, Software Safety, and Process Security, June 25-27,** Gaithersburg, Md. Cosponsors: IEEE Aerospace and Electronics Soc., IEEE Nat'l Capital Area Council. Contact Dolores R. Wallace, Nat'l Inst. of Standards and Technology, Gaithersburg, MD 20899, phone (301) 975-3340, e-mail wallace@swe.ncsl.nist.gov.

**Arith 10, 10th Symp. on Computer Arithmetic, June 26-28,** Grenoble, France. Cosponsors: ACM et al. Contact Jean-Michel Muller, Lab. LIP-IMAC, Ens. Lyon, 69364 Lyon Cedex 07, France, phone 33 (72) 72-8229.

## July 1991

**CAR 91, Fifth Int'l Symp. on Computer-Assisted Radiology, July 3-6,** Berlin. Sponsor: Tech. Univ. Berlin. Contact Heinz U. Lemke, Inst. for Tech. Computer Science, Sekr CG-FR3-3, Franklinstrasse 28-29, D-1000, Berlin 10, Germany, phone 49 (30) 314-73100; or Michael H. Rhodes, Toshiba America MRI, 280 Utah Ave., South San Francisco, CA 94080, phone (415) 875-2909.

**SIGGraph 91, July 30-Aug. 1,** Las Vegas. Sponsor: ACM. Contact Assoc. for Computing Machinery, 11 W. 42nd St., New York, NY 10036, phone (212) 869-7440.

## August 1991

**SSD 91, Second Symp. on Large Spatial Databases, Aug. 28-30,** Zurich, Switzerland. Contact H.J. Schek, Inst. für Information Systeme, Eth Zentrum, 8092 Zurich, Switzerland, phone 41 (1) 254-7240.

## September 1991

**VLDB 91, 17th Int'l Conf. on Very Large Data Bases, Sept. 3-6,** Barcelona, Spain. Sponsors: IEEE Computer Soc. Tech. Committee on Data Eng. et al. Contact Guy M. Lohman, IBM Almaden Research Center, Dept. K55, Bldg. 801, 650 Harry Rd., San Jose, CA 95120-6099, e-mail lohman@ibm.com (Internet), lohman@almaden (Bitnet).

**Compsac 91, 15th Int'l Computer Software and Applications Conf., Sept. 11-13,** Tokyo. Cosponsor: Information Processing Soc. of Japan. Contact Stephen S. Yau, Univ. of Florida, CIS Dept., Rm. 301, Gainesville, FL 32611, phone (904) 335-8006.

## October 1991

**IEEE Workshop on Visual Motion, Oct. 6-9,** Princeton, N.J. Contact Thomas S. Huang, Coordinated Science Lab, Univ. of Illinois, 1101 W. Springfield Ave., Urbana, IL 61801, phone (217) 333-6912.

**Fifth Software Eng. Inst. Conf. on Software Eng., Oct. 7-8,** Pittsburgh. Sponsor: SEI. Contact James E. Tomayko, SEI, Carnegie Mellon Univ., 4500 Fifth Ave., Pittsburgh, PA 15213-3890, phone (412) 268-6806, fax (412) 268-5758, e-mail jet@sei.cmu.edu.

**Workshop on Experimental Distributed Systems, Oct. 12,** Huntsville, Ala. Contact Raif M. Yanney, TRW, 1 Space Park, DH2/2328, Redondo Beach, CA 90278, phone (213) 764-6033.

**11th Symp. on Mass Storage Systems, Oct. 13-17,** Monterey, Calif. Sponsor: IEEE Computer Soc. Tech. Committee on Mass Storage. Contact Bernard T. O'Lear, NCAR, PO Box 3000, Boulder, CO 80307, phone (303) 497-1268.

**RIDT 91, Second Int'l Workshop on Raster Imaging and Digital Typography, Oct. 14-15,** Boston. Contact Robert A. Morris, Math. and Computer Science Dept., Univ. of Massachusetts at Boston, Harbor Campus, Boston, MA 02125-3393, phone (617) 287-6466, e-mail ridt91-request@cs.umb.edu.

**ICCD 91, IEEE Int'l Conf. Symp. on Computer Design, Oct. 14-16,** Cambridge, Mass. Cosponsors: IEEE Computer Soc. and IEEE Circuits and Systems Soc. Contact ICCD 91, IEEE Computer Soc., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

**16th Conf. on Local Computer Networks, Oct. 14-17,** Minneapolis, Minn. Cosponsor: IEEE Computer Soc. Technical Committee on Computer Comm. Contact James F. Mollenauer, 16th LCN Conf., Artel Communications, 22 Kane Industrial Dr., Hudson, MA 01749, phone (508) 562-2100, fax (508) 562-6942.

**Sixth Int'l Workshop on Software Specification and Design, Oct. 25-26,** Como, Italy. Contact Jean-Pierre Finance, CRIN, Campus Scientifique, BP 239 54000 Nancy, France, e-mail finance@loria.crin.fr.

**ITC 91, Int'l Test Conf., Oct. 28-Nov. 1,** Nashville, Tenn. Cosponsor: IEEE Philadelphia Section. Contact IEEE Computer Soc., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

## November 1991

**TAI 91, IEEE Computer Soc. Conf. on Tools for AI, Nov. 10-13,** San Jose, Calif. Contact Nikolaus G. Bourbakis, 4138 Moonflower Ct., San Jose, CA 95135, phone (408) 284-6494.

**ICCAD 91, IEEE Int'l Conf. on Computer-Aided Design, Nov. 11-14,** Santa Clara, Calif. Cosponsor: IEEE Circuits and Systems Soc. Contact ICCAD 91, IEEE Computer Soc., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

**Supercomputing 91, Nov. 18-22,** Albuquerque, N.M. Cosponsor: ACM. Contact Raymond L. Elliott, Computing and Comm. Div., MS B260, Los Alamos Nat'l Lab, Los Alamos, NM 97545; or Supercomputing 91, IEEE Computer Soc., 1730 Massachusetts Ave. NW, Washington, DC 20036-1903, phone (202) 371-1013.

## December 1991

**World Congress on Expert Systems, Dec. 16-19,** Orlando, Fla. Cosponsors: Int'l Assoc. of Knowledge Engineers et al. Contact World Congress on Expert Systems, c/o Congress Secretariat, Congrex (USA), Inc., 7315 Wisconsin Ave., Suite 404E, Bethesda, MD 20814, phone (301) 469-3355, fax (301) 469-3360.

The Seventh IEEE Conference on

# ARTIFICIAL INTELLIGENCE APPLICATIONS

February 24-28, 1991
Fontainebleau Hotel
Miami Beach, Florida

Se June Hong, General Chair
Tim Finin, Program Chair
Dan O'Leary, Tutorial Chair
Jeff Pepper, Publicity Chair

The Seventh IEEE Conference on Artificial Intelligence Applications (CAIA-91) is devoted to the application of artificial intelligence techniques to real-world problems. This year's conference will focus on both case studies and advances in AI techniques and principles that underlie knowledge-based systems and which enable ever more ambitious real-world applications. This conference provides a forum for such synergy between applications and AI techniques.

Tutorial sessions will be held Sunday, February 24 and Monday, February 25. The technical program sessions will be Tuesday, February 26 through Thursday, February 28.

**THE CONFERENCE KEYNOTE ADDRESS,** titled "Technology and People", will be presented by Eric Bloch, former director of the National Science Foundation.

**PLENARY TALKS:**
- Towards Intelligent Systems in the DoD, by Maj. Steve Cross, DARPA
- AI in Biology and Challenges of the Human Genome Project, by Bruce Buchanan, University of Pittsburgh

**INVITED SPEAKERS:**
- Application Projects at ICOT, by K. Furukawa, ICOT
- "Applying Commonsense" - Necessity or Oxymoron?, by Doug Lenat, MCC
- AI in the ESPRIT Program, by D. E. Talbot, ESPRIT

**PANELS:**
- Multi-Media and AI: Challenges and Opportunities
- Is Qualitative Physics Practical?
- AI in Design: User Perspectives
- When Does Truth Maintenance Pay Off?
- Generic and Consensus Reality Knowledge Bases and Their Use
- The Role of Standards in Knowledge Based Systems

**TUTORIALS:**
- Blackboard Applications
- User Modeling
- Model-Based Diagnosis
- Object Oriented Programming and Expert Systems
- Pattern Recognition and AI
- Constraint-Based Reasoning
- Case-Based Reasoning
- Expert Systems for Project Managers
- AI in Engineering Design
- Verification: Techniques and Solutions
- AI in Scheduling
- Integrating Knowledge-based Systems and Hypermedia

**PAPER SESSION TOPICS:** ■ Molecular Biology Applications ■ Innovative Database Technology ■ Knowledge Acquisition and Refinement ■ Rule Based System Theory ■ Visualization and Cooperative Systems ■ Hybrid Knowledge Representation Languages ■ Business/Management Decision Support ■ Image Understanding ■ Design and Manufacturing ■ Machine Learning ■ Design, Optimization and Decision Theory ■ Scheduling and Planning ■ Constraint/Belief Network ■ Dynamic Planning ■ Monitoring, Management and Uncertainty ■ Transportation Scheduling and Planning ■ Natural Language Processing.

**COST:** Take advantage of the lower registration fee being offered to everyone who registers before January 28, 1991. **Conference:** $245 for members, $310 for non-members, $90 for students. **Tutorial:** $150 for members and students, $185 for non-members (price is per tutorial). Rooms are available at the Fontainebleau Hotel at the special rate of $130, single and $145 double.

**FOR ADDITIONAL INFORMATION:** A complete advance program, including conference registration and hotel reservation forms, can be obtained from the IEEE Computer Society, 1730 Massachusetts Ave., N.W., Washington DC 20036; (202) 371-1013.

1951-1991
40 YEARS OF SERVICE
IEEE COMPUTER SOCIETY

IEEE
THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Editor: Guy Johnson, Department of Information Technology, Rochester Institute of Technology, 1 Lomb Memorial Drive, Rochester, NY 14623.

## *Modula-2 Programming: A First Course*

Edward D. Harter (Prentice Hall, Englewood Cliffs, N.J., 1990, 519 pp., $36)

This is a conventional first-course textbook in programming with Modula-2. In addition to the features of the language, it discusses random number generation, I/O, files, searching, sorting, and dynamic data structures. Most of the "systems programming" features of the language are not discussed (such as coroutines and anchored variables). There is a 10-page discussion of programming standards, but no mention of programming style.

The Modula-2 language was invented by Niklaus Wirth in 1978 as part of his Lilith workstation project. One of the self-imposed constraints of this project was that all software be written in a single language. Since no existing language was suitable, he invented a new one based on Pascal and Modula. Modula-2 benefits from 10 years of experience with Pascal and resembles that language closely. However, the syntax has been simplified somewhat, and some Pascal features have been discarded. The major addition to Pascal is the module, which resembles the package in Ada and the units found in many Pascal implementations. A number of low-level capabilities were added so that the operating system software could be written in Modula-2. These include the ability to address absolute memory locations and to receive interrupts.

Modula-2 was released in 1980 and has been very slowly supplanting its parent, Pascal. So far, Modula-2 has not been standardized; the "bible" for Modulans is Wirth's book, *Programming in Modula-2* (4th edition, Springer-Verlag, 1988). A draft for a proposed international standard for Modula-2 was released in late 1989. Harter's book is not tied to any particular Modula-2 implementation and does a good job of warning about possible incompatibilities with the student's system.

The production quality of the book is only fair. The listings are dot matrix printouts; these are very clear and readable. However, there are a moderate number of typos, such as "wisely used" instead of "widely used." The listings on a couple of pages are scrambled. Surprisingly, there are a number of misprints in the program listings themselves. There are also a few minor factual errors, such as an incorrect definition. On one page, the book claims that variable parameters are always passed by reference and value parameters are always passed by value; in fact, this is implementation-dependent. On another page, the book recommends declaring large array parameters as variables, even if they should be value parameters, to avoid the cost of copying them for the call. However, most implementations pass large structures by reference even if they are value parameters.

The book has several weak points. One is the absence of any graphics programs, a surprise in this age of bit-mapped displays. (There are not even any histograms printed with characters.) The examples are oriented towards data processing, and several times I thought I had stumbled into a Cobol course. Students are fascinated by seeing the computer draw pictures, and I think it would help the book to have a few such programs. (A possible supplement is Russell L. Schnapp's *Macintosh Graphics in Modula-2* (Prentice Hall, 1986.) Schnapp's book uses mostly turtle graphics, so it is not really Macintosh-specific.)

Another weakness is a poorly integrated presentation. Many topics are introduced and then never seen again; in particular, Chapter 1 seems dissociated from the rest of the book. As I was reading, I got the impression that the book was written according to some curriculum guide or syllabus, and a lot of miscellaneous topics were crammed in whether or not they fit naturally.

The book's greatest weakness is its examples. These are deliberately kept as simple as possible. While this works well in some ways, it teaches coding more than programming. Students never have to tackle anything very complicated, since the examples do not increase in sophistication as they work through the book.

The strong point of this book is its clear and extremely detailed explanations. Important notes and cautions are set off in boxes. The descriptions of operations on data structures are especially good; they have many pages of diagrams, such as how linked lists tie together.

Despite the book's weaknesses, I think it is adequate for a first course. The instructor must supplement it in various ways; in particular, he or she should introduce some fairly complicated examples that can be worked on for several weeks so the students will gain a little insight into large program development. The ordering of the advanced topics is sometimes peculiar, and the instructor will probably want to alter this. For example, sorting precedes searching, and hash searches are described before sequential searches. Recursion has been relegated to an appendix, while the body of the book has implementations of Quicksort and tree traversal using explicit stacks. This throws away two excellent opportunities to show the value of recursion, and nobody would implement these algorithms with explicit stacks unless he or she was writing in Fortran.

This book would not be suitable for individual study. People who already know programming would not want to wade through it just to learn Modula-2. People who don't know programming could learn the constructs of Modula-2, but they would not learn how to attack a programming problem and structure the solution.

Allen Stenger
Gardena, California

## *Great Ideas in Computer Science: A Gentle Introduction*

Alan W. Biermann (The MIT Press, Cambridge, Mass., 1990, 446 pp., $27.95)

What are the great ideas in computer science? What exactly do computer scientists do? And how do you introduce technical concepts and activities into a computer literacy course teaming with liberal arts students? These are among the questions *Great Ideas in Computer Science* sets out to answer.

The first third of this book discusses Pascal programming. After an initial chapter of basic information on running a first computer program, the book progresses with detailed analyses of sample Pascal programs. The reader is expected, workbook style, to carry out the book's sample programs and exercises. It would be difficult to follow the text without actually sitting down at a computer and compiling the sample programs. The reader is quickly challenged, as increasingly more complex Pascal constructs and programming techniques follow.

*Great Ideas in Computer Science* challenges its readers with Pascal programming's most profound principles. Following brief outlines of procedures and functions, for example, the discussion focuses on every possible means of passing values into these. Biermann even uses an array as the value passing parameter. Referencing values into procedures and functions is an arduous learning process for liberal arts students and beginning programmers, and passing values from an array can be especially perplexing. Biermann also discusses recursion in some depth. This principle is often bypassed in freshman engineering-level structured-programming courses because of its inherent difficulty. In short, almost no characteristic of Pascal programming is overlooked in this book.

One of the best sections of this book is "Electric Circuits." This chapter skillfully explains how to configure simple battery circuits to generate complex Boolean logic functions. The discussion includes relays, transistors, and electromagnets and uses well-drawn diagrams and examples to demonstrate how binary circuits are created. This rudimentary computer circuitry provides a natural transition into microprocessor technology.

VLSI technology, the key to computer architecture, is another self-contained chapter. Mask layout, transistor sizing, transistor delay circuitry, current gain, symbolic layout, and circuit design computer simulation are some of the topics covered in this section.

**_Great Ideas in Computer Science_ challenges its readers with Pascal programming's most profound principles.**

The chapter on machine architecture examines the operation of the P88, part of the Intel 8088 microprocessor. It focuses on the interior workings of the P88 and its assembly language. A prior explanation of assembly language might have made the P88 more understandable to the book's intended readership. For example, an introduction to basic computer organization through the von Neumann machine might have been gentler. This simple diagram could have enhanced the reader's understanding of the processor's interactions with other computer units.

Chapters on language translation, program execution time, and parallel computation continue to illuminate difficult concepts with intricate and well-written explanations. An overview of connectionist machines, probably the most enigmatic computers receiving contemporary scrutiny, provides an interesting sidelight. A section on problems not computable by computer and a final section on artificial intelligence round out this challenging book.

In summary, *Great Ideas in Computer Science* is very readable and offers many finely drawn diagrams and sketches. The logical organization of its subject matter provides good continuity with each idea building on its predecessor. Every chapter has a bibliography listing major books related to its theme. The thoughtful programming exercises provide interesting and challenging problems. The subject matter selection succeeds in revealing what computer scientists do, what their workaday lives are like, and what general problems confront computer science today.

Yet, it is the very originality of this book that contributes to its most significant problem. The complex topics are not routinely found in introductory computer literacy texts. Because of this book's depth and technical treatment, its intended audience of liberal arts students will not discover the gentle introduction suggested by the book's subtitle. At the same time, the challenging content of *Great Ideas in Computer Science* makes an interesting overview of the computer science field sure to be welcomed by the beginning technical or computer science student. In this regard, the book offers a real contribution.

Don Bissell
Harvest Lane Associates
Wells, Maine

## *Mind Children: The Future of Robot and Human Intelligence*

Hans Moravec (Harvard University Press, Cambridge, Mass., 1988, 214 pp., paperback $8.95, hardcover $18.95)

In this extremely well written book, the author provides a brilliant account of the historical development of computers, artificial intelligence, and robotics.

Industrial robots have undergone many developments since the arrival of the first Unimation machine in 1962. Applications are widespread and growing. The robot's advantages of flexibility, reprogrammability, tirelessness, and hardiness have come to be appreciated by industrialists. Even the layman realizes that today's industrial robot, unlike the tin marvel of science fiction, has a real and useful role to fulfill.

However, despite many developments in the associated technologies, the industrial robot capable of sensing and reacting to the external environment is still in its infancy. Robots have been and continue to be shaped in the image of man, and the second generation is seen to be a further step in this direction. However, anthropomorphism is a constraining influence, and new perspectives are needed. Perhaps we should endeavor to see beyond the principles of replacing a human by a human-like robot. It's high time for a new look at manufacturing processes — particularly in assembly and product design for assembly. Flexible manufacturing systems (FMS) involving squat, multiarmed robots with second-generation capabilities, interacting with machine tools under integrated computer control, offer great potential. Furthermore, as robot capabilities are enhanced to meet the challenge of the more difficult industrial tasks, applications outside the factory will become more feasible, lending credence to the present research on mobile robots capable of operating in unstructured environments, such as homes and hospitals.

Robotics is a multidiscipline activity involving mathematicians, physicists, electrical engineers, mechanical engineers, and computer scientists. It is an ideal subject for illustrating the systems approach, and as such it offers a useful means for broadening a learner's perspectives.

There are many instances where it is necessary to use mobile robots. In flexible machining systems, in addition to developing flexible machines for assembly, machining, etc., it is equally important that there are flexible means of transportation among the various production processes.

Fixed equipment, such as rollers, belts, and overhead conveyors, is inflexible and can be expensive to change if machine layouts and transportation routes have to be varied. In recognition of this, automated guided vehicle systems have become increasingly popular over the last 30 years. Such systems employ computer-controlled skips or skips guided by embedded wires. But, greater flexibility can be achieved if instead of restricting the vehicle's movement to a set of guideways, it can be taught the route by an operator like any other robot.

In addition to FMS, there are many other situations where mobile robots could be quite useful; for example, the domestic robot to help with housework; the night watchman robot that can go right to the heart of the fire; and the patient-care robot that can relieve the nurse of hard physical labor such as lifting, holding, and carrying patients or handicapped children.

*Mind Children* is full of ideas and information, and it is extremely easy to read. The book, which consists of six chapters and three appendices, could be used as required reading for students in engineering and computer science at the introductory level. Also, this book is useful for anyone who wishes to keep abreast of recent technological developments in robotics and artificial intelligence.

Sheo G. Misra
Wilkes University
Wilkes Barre, PA

## *Strategies for Real-Time System Specification*

Derek J. Hatley and Imtiaz A. Pribhai (Dorset House, New York, 1988, 386 pp., $51)

This book — which is somewhat broader than its title suggests — describes a methodology for expressing requirements for software-based systems and expressing the structure of these systems. It combines recipes describing what is to be solved with how to solve it. It goes far beyond requirements specification and says how to design systems. Going deeply into the design process, it says not only what is to be structured but also how it is to be done. Since there is nothing wrong with such an approach, I would only warn that this understanding of the development process is a little bit different from more common IEEE-endorsed terminology (see IEEE Std. 1074, Software Life-Cycle Processes, for example, where requirements documents are called specifications, and design documents are called descriptions).

**This book goes far beyond requirements specification and says how to design systems.**

The preface says that this book is "for a wide range of audience." I would be more specific and add that it is for everyone who wants to learn the methods presented, including system specifiers, designers, software engineers, system engineers, analysts, maintainers, and project managers.

What will they gain from reading this book? It presents two detailed, fairly cohesive methods for specifying systems (Parts II and III, "The Requirements Model" and "Building the Requirements Model") and for designing them (Parts IV and V, "The Architecture Model" and "Building the Architecture Model"). The methods largely follow two older approaches to structured design, presented in widely recognized books, authored by E. Yourdon and L.L. Constantine (*Structured Design*, Prentice Hall, 1975) and by Tom De Marco (*Structured Analysis and System Specification*, Prentice Hall, 1978). In particular, the book is a good continuation of the structured analysis and design methods of De Marco, who also wrote a foreword to the present book.

What is the essence of these methods? In the first area, building the requirements model, the book includes the development of data context and control context diagrams, data flow and control flow diagrams, process and control specifications, timing specifications, and a requirements dictionary. The second area includes the development of the following for building the architectural model: context diagrams, flow and interconnect diagrams, module and interconnect specifications, and a dictionary. The rest is left to the implementors.

As the title implies, the book covers strategies, that is, methods rather than techniques. Although the methods and their presentation are rigorous and systematic, they are not formal. The only theoretical base is finite state machines. Certainly, the methods could be made more formal, thus leading to the development of certain techniques, for example, how to combine the data flow and control flow diagrams. As a direct consequence, the book does not cover tools, even though the authors suggest that the methodology can be viewed as a tool kit, which it cannot. However, tools for these strategies are being extensively developed by commercial companies, and references are included in the bibliography.

It is not my intention to assess the methods themselves or make direct recommendations about their use. I assess the book's treatment of the methods, and in that regard, it is excellent. It describes these strategies more than adequately; moreover, it does so in a structured and clear way.

What about the book's relation to real-time issues? It is hard to clearly answer this question and assess the book in this respect without applying the strategies. Definitely, the book addresses the problem. Real-time requirements are covered by timing specification, at least. Is the treatment of real-time aspects intrinsic to the method, or are real-time related properties only added to normal conventional requirements? Certainly, this is related to the question, does this method find what is real-time specific? Without using this methodology, I cannot easily determine if the strategies presented capture the intricacies of real-time systems. Discussions in the book about timing properties of systems, although extremely interesting, did not convince me that the methods own something special related to real time. The facts are more obvious and encouraging, however. The methods proved extremely useful in designing a real-time embedded avionics system. They are in widespread use in other industries as well. This is confirmed by three case studies that end the book.

In similar publications, authors often try to make a title wider than contents, hoping to attract a broader audience. Here, on the contrary, the book's content is wider than its title. The book is not limited to the specification of requirements, because the authors' understanding of this term includes the specification of the design. Furthermore, it is not necessarily confined to real-time systems, as the methods may deal with a wide range of other applications as well. Although the book is somewhat limited, since it presents only one view and one solution of the problem (this is its real strength, on the other hand), I would recommend it for college use as a complementary textbook on design in software engineering and real-time systems courses.

Janusz Zalewski
Southwest Texas State University

## Current contents CS MAGAZINES

### *November 1990 IEEE Software*

### November 1990 *IEEE Computer Graphics and Applications*

**For subscription information, circle number 200 on the reader service card.**

# CAREER OPPORTUNITIES

**RATES:** $12.00 per line, (ten lines minimum). Average five typeset words per line, eight lines per column inch. Add $10 for box number. Send copy at least one month prior to publication date to: **Marian B. Tibayan, Classified Advertising, COMPUTER Magazine, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1264; (714) 821-8380; fax:(714) 821-4010.**

In order to conform to the Age Discrimination in Employment Act and to discourage age discrimination, *COMPUTER* may reject any advertisement containing any of these phrases or similar ones: "...recent college grads...," "...1-4 years maximum experience...," "...up to 5 years experience," or "...10 years maximum experience." *COMPUTER* reserves the right to append to any advertisement without specific notice to the advertiser. Experience ranges are suggested minimum requirements, not maximums. *COMPUTER* assumes that since advertisers have been notified of this policy in advance, they agree that any experience requirements, whether stated as ranges or otherwise, will be construed by the reader as minimum requirements only.

## UNIVERSITY OF SOUTH FLORIDA
**Chairperson**
**Department of Computer Science and Engineering**

The Department of Computer Science and Engineering at the University of South Florida invites nominations and applications for the position of Chairperson. The Department offers Bachelor's degrees in Computer Science (accredited by CSAB) and in Computer Engineering (accredited by ABET), Master's and Ph.D. degree programs. The current faculty size is 15, with several new positions to be added over the next two years. The four broad areas of research emphasis chosen by the faculty are

- Computer Architecture/VLSI Design & Test,
- Computer Vision/Graphics/Image Processing,
- Artificial Intelligence/Expert Systems, and
- Software Engineering.

The Department is relatively young and ambitious, with a rapidly expanding research program. An experienced individual is sought as Chairperson to lead the Department to a position of national/international recognition.

The Department shares a new 12 million dollar building with the Department of Electrical Engineering. The Department research network includes a substantial number of SUNs and VAXes, an INTEL 2/386 hypercube, a variety of specialized graphics and image processing equipment, and a number of other resources. Additional computing resources are available on the College computing network and the University network.

The University of South Florida is the second largest university in the State of Florida, and the forty-second largest in the nation, with an enrollment of well over 30,000. USF occupies a 1,700-acre campus in the city of Tampa, one of the largest and fastest growing metropolitan areas in Florida, offering a wide variety of cultural, entertainment, sports and outdoor activities. The quality of life is excellent and the cost of living is moderate. The area near the University is experiencing dramatic growth in high-technology industry and medical facilities. The faculty of the Department have interactions with a number of companies in the area, including AT&T, E-Systems, GTE Data Services, Harris, Hercules, Honeywell and IBM. A PBS public-TV channel is located on campus; other television links tie the University to industry and remote campuses.

Applicants should send a resume and the names of three references to the Department Chairperson Search Committee, Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620.

The University of South Florida is an equal opportunity and affirmative action employer.

## International Computer Science Institute
**Berkeley, California**
**Postdoctoral and Visiting Positions**

The International Computer Science Institute announces the availability of postdoctoral and visiting appointments for 1991 and beyond. The *Institute* is a non-profit basic research organization physically near and loosely affiliated with the University of California at Berkeley. Postdocs will work with the *Institute* and UCB faculty and staff on current projects. More senior visiting researchers can propose projects of any length and character. *ICSI* conducts research in several areas of parallel and distributed computation including theory, realization and applications of massive parallelism and the design of multimedia distributed systems and very high-speed networks.

Applicants should submit a resume, the names and addresses of three references, selected publications and a one-page research plan as soon as possible to:
Jerome A. Feldman, Director
International Computer Science Institute
1947 Center Street, Suite 600
Berkeley, CA 94704-1105 U.S.A.
FAX (415) 643-7684
Internet/CSnet:info@icsi.berkeley.EDU

In order to save time, applicants should ask their references to write directly to the above address so that letters of evaluation reach *ICSI* at about the same time as the other application materials. Selection will be based on overall qualifications and compatibility with *ICSI* research plans. Applications from citizens of *ICSI* sponsor nations are especially welcome.

ICSI is an Equal Opportunity Employer.

## UNIVERSITY OF CALIFORNIA, SANTA BARBARA
**Department of Computer Science**

The Department of Computer Science at the University of California at Santa Barbara invites applications for *junior and senior* tenure-track faculty positions. Senior applicants should possess distinguished research records and the ability to attract research funding, while junior candidates must demonstrate exceptional promise.

Applicants will be considered in all areas of Computer Science, although the department is currently attempting to achieve strengths in the areas of software systems, computer systems modeling and analysis, algorithms and complexity, parallel and distributed computing, scientific computation, and machine intelligence. Resources, tailored to the needs of successful applicants, will be available for state-of-the-art laboratories for research and instruction. Responsibilities include a strong emphasis on research, supervision of graduate students, teaching graduate and undergraduate courses, participation in departmental and university committees.

All applicants should hold a doctoral degree in Computer Science or a related field. Appointments are scheduled to begin in 1991-92. Unfilled positions will remain open until filled. Send resume and names of at least 4 referees to:
Recruitment Committee
Department of Computer Science
University of California
Santa Barbara, CA 93106

Proof of U.S. citizenship or eligibility for U.S. employment will be required prior to employment *(Immigration Reform & Control Act of 1986)*. The University of California is an Equal Opportunity/Affirmative Action Employer.

## ASTEM, KYOTO, JAPAN
**Advanced Software and Mechatronics Research Institute**

The newly established ASTEM, Kyoto, Japan, invites applications for research associate and senior research positions.

Speciality areas include, but are not limited to, Software Engineering, Knowledge Base, Computer Graphics, and CAD/CAM/CIM. For a research associate position, a candidate should be highly qualified and hold an MS in CS, EE, or related fields; for a senior research position a PhD is normally required.

ASTEM provides an excellent environment for research in Software Engineering and AI with unique Japanese software and the state of the art workstations.

ASTEM is located in Kyoto, the ancient beautiful city of Japan.

Send resume to: Mr. A. Kamei,
ASTEM, Kyoto Research Park, 17 Chudoji, Minami-machi, Shimogyo, Kyoto 600 JAPAN
or call (213) 544-1103.

## UNIVERSITY OF PITTSBURGH
### Department of Computer Science

The Department of Computer Science invites applications for one or more tenure-track faculty positions at the assistant professor level areas of interest are Systems, Data Base, and Artificial Intelligence. The starting date will be September 1, 1991. Responsibilities include research, supervision of graduate student research (Ph.D. and M.S.), and graduate and undergraduate teaching. Candidates should have a Ph.D. in computer science and a strong interest in both research and teaching.

The Department currently has twenty-three full-time faculty members and supports strong graduate and undergraduate programs. Departmental resources include an excellent research library and extensive computing facilities including a network of SUN and Xerox 1100-series (Dandelion) workstations, a VAX 11/780 (under BSD UNIX), an Intel iPSC/2 hypercube, a variety of micro-computers, and several graphics systems. The research systems are accessible via the Department's Ethernet-compatible LAN. Convenient access is also provided to the extensive general computer facilities of the University as well as to other networks (e.g., ARPANET, CSNET). The Department operates the *Center for Parallel, Distributed and Intelligent Systems* (CPDIS) to provide an environment for innovative research in computer science. Since the University of Pittsburgh is a founding member of the Pittsburgh Supercomputing Center and an affiliate member of the Software Engineering Institute, the Department of Computer Science has access to the Cray X-MP/48 of PSC and the software engineering expertise at SEI.

Please send your resume to: Dr. Rami Melhem, Chair of Faculty Search, Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260.

Pitt is an equal opportunity/affirmative action employer and especially encourages women and members of ethnic minorities to apply.

## UNIVERSITY OF CONNECTICUT ASSISTANT PROFESSOR COMPUTER SCIENCE & ENGINEERING

The Department of Computer Science and Engineering at the University of Connecticut is seeking an outstanding applicant with a demonstrated research ability to fill one anticipated tenure-track faculty position at the Assistant Professor level beginning with 1991-1992 academic year. The University is located in a rural area in Northeast Connecticut within easy driving distance of several major metropolitan areas. The department offers B.S.E., M.S., and Ph.D. degrees in Computer Science within the School of Engineering. Applicants with Ph.D. in Computer Science, Computer Engineering or equivalent areas are invited to submit resumes and three letters of reference to: Chair, Search Committee, University of Connecticut, Computer Science and Engineering Department, U-155, 260 Glenbrook Road, Storrs, CT 06269-3155. AA/EOE. (Search #1A115).

## UNIVERSITY OF CALIFORNIA AT IRVINE
### Faculty Positions in Computer Science

The Department of Information and Computer Science (ICS) is actively recruiting faculty AT ALL LEVELS. We have dynamic research groups in the areas of computer systems design, parallel processing, artificial intelligence, computer networks and distributed processing, software, social and managerial analysis of computing, and theory. We are continuing to build on these areas of strength. We are also interested in developing new strength in computational biology, integrated systems, computer-supported cooperative work, databases, design tools, and model-based reasoning. We will sympathetically review applications from very strong candidates in all areas of computer science.

We are looking for new faculty with strong research records who would thrive in a highly productive but friendly setting, and who would like to join us in exploring the nature of computing, broadly defined. We specially encourage application from exceptionally distinguished candidates for senior positions.

The ICS Department is an independent academic unit reporting to the Executive Vice Chancellor. ICS faculty emphasize core computer science as well as research in emerging areas of the discipline, with effective interdisciplinary ties to colleagues in neurobiology, cognitive science, management, engineering, and the social sciences. The department currently has 30 full-time faculty positions and over 120 Ph.D. students, with major support from the administration to expand and to strengthen the research environment.

Annual research funding from contracts and grants from agencies such as DARPA, NSF, and ONR, currently total over $6.5 million. In 1986 the software group was awarded a Coordinated Experimental Research (CER) grant from the National Science Foundation. This support has fostered the creation of a Research Laboratory for software engineering, in which major studies of the development and evaluation of software technology are undertaken. High quality research has also fostered the creation of a Research Laboratory for computer systems design that deals with methodologies and tools for the design of complex computational systems. A third Research Laboratory, in Artificial Intelligence, is planned.

Department equipment includes approximately 175 workstations, primarily Sun-3's and Sun-4s. Two large multiprocessor Sequents and a Hypercube are available, as well as approximately 300 Macintosh Plus's and II's. All our major workstations and computers are tied together with networks, which are gatewayed to the campus network, and from there, to regional, national, and international networks (Darpa Internet, CSnet, Bitnet, etc.). In addition, department members have access to campus-wide computing resources as well as regional supercomputer access.

UC-Irvine is located in Orange County, three miles from the Pacific Ocean near Newport Beach, and approximately halfway between Los Angeles and San Diego. The campus is situated in the heart of a national center of high-technology enterprise. It is growing rapidly and offers exciting professional and cultural opportunities. Salaries and benefits are competitive. Special housing assistance is available from the university, including newly built, for-sale housing within short walking distance from the Department.

Send resumes and names of four references to:

Chair, Faculty Recruiting Committee
Department of Information and Computer Science
University of California-Irvine
Irvine, CA 92717

The ICS Department has several vacant positions and application screening will begin immediately upon receipt of curriculum vitae. Maximum consideration will be given to applications received by January 31, 1991.

The University of California is an Affirmative Action/Equal Opportunity Employer. The Department of ICS is particularly interested in receiving applications from women and minority candidates.

## THE UNIVERSITY OF AUCKLAND NEW ZEALAND
### Two Chairs in Computer Science

The University of Auckland seeks to appoint two qualified Computer Scientists who have research and teaching skills that will enable them to make significant contributions to its rapidly developing Department of Computer Science.

The University, with over 16,000 students, holds a premier position and is sited in the heart of New Zealand's largest city. Auckland, City of Sails, is the international gateway to New Zealand, the major industrial, commercial and cultural city in the country, and offers an exceptional range of lifestyles and recreational activities.

Our Department of Computer Science, now ten years old, has 370 equivalent full-time students. In the past the department has led the way with the use of modern computing technology in teaching. Now it is poised on the brink of a second period of development following a recent review of the position of computing disciplines within the university. The new chairs will provide leadership for this new development and the associated expansion of the department's resources.

Applications are welcomed from those who believe they are qualified for these challenging positions. The successful applicants will be expected to have advanced qualifications, accomplished research records and a demonstrated history of teaching, administration and liaison with industry and commerce. The new chairs will be encouraged to foster relations with the business sector, including engagement in consultancy activities.

The precise conditions of appointment are subject to negotiation. Further information including standard Conditions of Appointment and Method of Application are available from the Assistant Registrar (Academic Appointments), University of Auckland, Private Bag, Auckland, FAX 64 (9) 799 317. Applications should be forwarded by the closing date **31 JANUARY 1991.**

The University of Auckland
An Equal Opportunity Employer

**WASHINGTON STATE UNIVERSITY**

The Department of Computer Science has recently merged with the Department of Electrical Engineering, and we seek to strengthen our capabilities in selected areas of Computer Science. Applications are invited for full-time tenure-track positions at Assistant Professor, Associate Professor and Full Professor levels. Responsibilities include undergraduate and graduate teaching and the initiation, conduct and supervision of research. Minimum qualifications include a Ph.D. degree in Computer Science or closely allied field and a demonstrated potential for research. Candidates for the higher ranks must have proven records of accomplishment as evidenced by publications and sponsored research. Areas of primary interest are artificial intelligence, database systems, software engineering and parallel and distributed computing. Screening will begin immediately and continue until all openings are filled. Positions are available starting January 1, 1991 and August 16, 1991.

To apply, send a resume and the names and addresses of at least three references to: Dr. Yacov Shamash, Chairman, Department of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752. WSU is an EA/AA educator and employer. Protected group members are encouraged to apply.

**COLGATE UNIVERSITY**
**Hamilton, New York 13346**
**Department of Computer Science**
**(315) 824-1000, ext. 719**

We invite applications for two anticipated positions, pending administrative approval: one a tenure track position at the rank of assistant professor; the other a one-year leave-replacement position. Candidates should have or be near completion of a Ph.D. in computer science. Strong candidates in any subfield of the discipline will be considered.

Colgate is a quality liberal arts college with a first-rate computer science program. The department has five faculty with the following research interests: theory of computation and programming language semantics, computational complexity and algorithms, temporal reasoning in natural language processing, graphics and chaos, and discrete event simulation on parallel computers. The Computer Science Department has an introductory lab equipped with sixteen PCs, and an upper-level/research lab with the following equipment: a network of seven NeXT workstations, a VAX 750 running BSD 4.3 Unix, four 17-node transputer-based parallel computers, and PC/AT or NeXT workstations in every faculty office. The faculty offices and laboratory machines are connected on an ethernet. We are members of both CSNet and BitNet.

Applicants should send a resume and the names of three references to: Chris Nevison, Chairman, Department of Computer Science, Colgate University, Hamilton, NY 13346. We will consider all applications received by January 1, 1991, and applications received thereafter until the positions are filled. EO/AAE.

**COLUMBIA UNIVERSITY**
**Department of Computer Science**

We are looking for several exceptional people to join our faculty. Tenure-track positions are available at all ranks in all areas. Applicants in hardware are particularly encouraged.

Our department of nineteen tenure-track and three teaching faculty emphasizes research, and attracts excellent Ph.D. students, virtually all of whom are fully supported. Departmental facilities include many advanced workstations, high-performance servers, and graphic systems, plus state-of-the-art systems designed and built at Columbia for vision, robotics, parallel computation, networking and distributed computing. We are within an hour's drive of the research laboratories of IBM, AT&T, Bellcore, Siemens, Philips, NYNEX, and other leading industrial companies.

Columbia University is one of the oldest and most prestigious universities in the United States, and New York City is one of the cultural, financial, and communications capitals of the world. The department is housed in its own building, and in 1992 we will acquire additional space and facilities in the interdisciplinary Center for Engineering and Physical Science Research now under construction. University-subsidized housing and parking is readily available.

Candidates for assistant professor should exhibit exceptional research promise, while those seeking a more senior position should have an outstanding record of research achievement. Interest and ability in teaching undergraduates and graduates is necessary. Send resume and the names of at least three references to: Prof. John R. Kender, Faculty Search Chairperson, Department of Computer Science, Columbia University, New York, New York 10027.

Columbia University is an Equal Opportunity/Affirmative Action Employer. We encourage applications from women and minorities.

**COMPUTER SPECIALIST**

Plans, organizes, directs and coordinates data processing services for State Vocational-Technical Schools and other State Schools in region. Performs systems analysis and programming utilizing VAX/VMS and database administration utilizing ORACLE relational database system. Trains and supervises technical and non-technical personnel. Confers with school directors and other administrative personnel concerning data processing needs and utilization of computer resources. Selects, customizes and maintains software utilized for instructional and administrative functions.

Requirements: B.S. or higher Degree in Computer Science and two years experience as Computer Systems Programmer utilizing VAX, VMS, DOS and ORACLE database.

$27,069.00 per year; 7:30 a.m. - 4:00 p.m.; 40 hours per week.

Must have proof of legal authority to work in the United States.

Contact Louisiana Office of Employment Security, 2900 Dowdell St., Shreveport, LA 71133 Job Order #833452.

**UTAH STATE UNIVERSITY**

Applications are invited for all faculty levels in Computer Science. Qualifications include a doctorate in Computer Science or a closely related field; a strong commitment to teaching at both the undergraduate and graduate levels; and a similar commitment to research. Specializations of particular interest include Parallelism, Software Engineering, and Artificial Intelligence.

The department currently offers B.S. and M.S. degrees in Computer Science. Successful candidates would be expected to assist in the development of a Ph.D.

The university is located in a beautiful mountain valley with easy access to recreational and cultural activities.

Send resumes and names of 3 references to: Gregory Jones, Computer Science, Utah State University, Logan, Utah, 84322-4205. Positions will open January and remain open until filled. U.S.U. is an EO/AAE employer.

**UNIVERSITY OF WASHINGTON**
**Department of Computer Science and Engineering**

The Department of Computer Science and Engineering at the University of Washington expects to have one or more tenure-track openings starting in the 1991-92 academic year. We seek outstanding applicants who add to our existing research strengths, particularly in programming languages, compilers, graphics, hardware/computer engineering, and software engineering, or who bring significant new research strength to our department.

A moderate teaching load allows time for quality research and close involvement with students. We expect applicants to have a strong commitment to both research and teaching, and an outstanding record of research for their level.

The department may also have visiting positions that would require both teaching and research. It may be possible to hold these for portions of the 1991-92 academic year.

Interested applicants should send a letter of application, a resume, and the names of four references to Faculty Recruiting Committee, Department of Computer Science and Engineering FR-35, University of Washington, Seattle, Washington 98195. Candidates are encouraged to apply as early as possible.

The University of Washington is an Affirmative Action/Equal Opportunity Employer. The Ph.D. is required for these positions.

**INDIAN INSTITUTE OF TECHNOLOGY, KANPUR**

Faculty positions in all areas of Computer Science are available. Special requirements: Computer Architecture and Systems. Send resume and names of three referees to the Head, Department of Computer Science & Engineering, Indian Institute of Technology, Kanpur—208 016 (India).

To expedite, request your referees to send recommendation letter to us at an early date.

## STATE UNIVERSITY OF NEW YORK AT BINGHAMTON
**Department of Computer Science**
**The Watson School of Engineering**

The State University of New York at Binghamton invites applications for tenure-track positions in the Department of Computer Science beginning August 1991. Positions are available at junior and senior levels and the salary is competitive. Preferred areas of specialization include distributed/parallel systems, databases, software engineering, and artificial intelligence, but applicants in all areas of Computer Science will be considered. Applicants must have a Ph.D. in Computer Science or a related area, and possess a strong commitment to research and teaching.

The Department has established Ph.D. and M.S. programs, and an accredited B.S. program. High technology computer-oriented companies in the local area such as IBM, G.E., Link Flight Simulation, and Universal Instruments provide opportunities for industrial collaboration.

Send nominations or applications including a resume and the name of three references to Professor Sudhir Aggarwal, Chairman, Department of Computer Science, The Watson School, State University of New York at Binghamton, P.O. Box 6000, Binghamton, New York 13902-6000. Applications received by January 15, 1991 will receive first consideration.

The State University of New York at Binghamton is strongly committed to affirmative action. We offer access to services and recruit students and employees without regard to race, color, sex, religion, age, disability, marital status, sexual orientation or national origin.

## OREGON GRADUATE INSTITUTE OF SCIENCE AND TECHNOLOGY

Would you like to work in an academic environment with an active graduate education program, but with no undergraduate teaching responsibilities? A place that encourages serious research by providing strong administrative support and excellent facilities? If so, consider joining the growing faculty of the Oregon Graduate Institute's Department of Computer Science and Engineering.

We seek both senior and junior faculty colleagues with experience in graduate education and ambitious research goals. Technical areas of particular interest include: scientific and engineering databases, distributed and concurrent computing systems, software specification and derivation, artificial neural networks and speech recognition.

OGI is located in Portland, one of the most affordable of the West Coast's beautiful cities. Portland's relaxed life style offers a setting in which both your family and your research can thrive.

For more information about OGI, please address inquiries to: Professor Richard B. Kieburtz, Chairman, Department of Computer Science and Engineering, Oregon Graduate Institute, 19600 NW von Neumann Drive, Beaverton, OR 97006, (503) 690-1150, csedept@cse.ogi.edu.

OGI is an Equal Opportunity Employer.

## UNIVERSITY OF PENNSYLVANIA

The University of Pennsylvania invites applications for faculty positions in the Department of Computer and Information Science, effective July 1, 1991. Outstanding candidates in the areas of computer graphics, scientific visualization, artificial intelligence, computer vision and programming languages will be given priority.

Applications (including the names of at least three references) should be submitted to Professor Bonnie Lynn Webber, Chair—Faculty Search Committee, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104-6389.

(The University of Pennsylvania is an Affirmative Action/Equal Opportunity Employer).

## CLEMSON UNIVERSITY
**Head, Department of Computer Science**

Clemson University invites nominations and applications for the position of Head, Department of Computer Science. The successful applicant is expected to have an earned Ph.D. in Computer Science or closely related field, and a record of excellence in research and in teaching at both the undergraduate and graduate level. Experience in departmental or university administration is highly desirable.

The Department of Computer Science, part of the College of Sciences, offers a CSAC/CSAB accredited BS program in computer science, a BS program in computer information systems, and established MS and Ph.D. programs in computer science. A full-time faculty of 22 supports approximately 300 undergraduate students, 75 MS students and 25 Ph.D. students. Computing facilities are provided by both the university and the department with excellent access to all systems through workstations in faculty offices, public access clusters and dedicated laboratory machines for specialized courses.

Clemson University, a land grant institution with 16,000 students, is located in the northwest corner of South Carolina, in the foothills of the Blue Ridge Mountains. Its 20,000 acre campus is adjacent to Lake Hartwell, which forms a boundary between South Carolina and Georgia. Approximately two hours on interstate highways from Atlanta, GA and Charlotte, NC, the University is located in the town of Clemson, SC, a small city with a population of 10,000. Quality of life, with outstanding opportunities for outdoor activities, is excellent while the cost-of-living is well below most areas of the country.

Qualified applicants should submit a resume with names and addresses of at least three references to:

Dr. John C. Peck, Chair
Departmental Search Committee
Department of Computer Science, Slot A
Clemson University
Clemson, SC 29634-1906
EMAIL: *Peck@CS.Clemson.Edu*

Selection of candidates for interviews will begin in February 1991. Clemson University is an Equal opportunity/Affirmative Action Employer. Women and minorities are encouraged to apply.

## STATE UNIVERSITY OF NEW YORK AT BUFFALO
**Department of Computer Science**
**Faculty Positions**

The Department of Computer Science is seeking candidates for faculty positions at junior or senior levels. Junior-level applicants must show excellent research promise, and must have completed all requirements for the Ph.D. degree in computer science or a closely related field before assuming duties. Candidates for senior positions must have an established research reputation.

The Department currently has 15 tenure-track faculty, 9 additional faculty, and 140 graduate students. Primary research areas include: artificial intelligence, complexity theory, computer vision, numerical linear algebra, parallel algorithms, programming languages, systems and VLSI. Department members are actively engaged in interdisciplinary research programs in the Advanced Scientific Computing Graduate Group, the Cognitive Science Center, the Vision Graduate Group, and the NSF National Center for Geographic Information and Analysis. Departmental computing facilities include a network of workstations, hypercubes, Symbolics, an Encore Multimax and several image processing/graphics systems. The department is expanding, with additional faculty lines committed annually. Salaries are competitive.

Applications should include a letter and a curriculum vitae. Applicants should arrange to have four letters of reference sent directly from their referees to: Dr. Deborah Walters, Chair of Search Committee, Department of Computer Science, 226 Bell Hall, SUNY at Buffalo, Buffalo, NY 14260. For full consideration applications should be received by January 20, 1991.

SUNY is an Equal Opportunity/Affirmative Action employer.

## THE UNIVERSITY OF TEXAS AT ARLINGTON

The Department of Computer Science Engineering at The University of Texas at Arlington invites applications for tenure-track or visiting faculty positions in all areas of computer science or computer engineering. Applicants with expertise relating to reliable real-time distributed systems, telecommunications software, object-oriented systems, scientific visualization, knowledge-based systems, or parallel processing will be given preference. Rank is open. An earned doctorate or equivalent and a commitment to teaching and scholarly research are required. Openings are expected for January and September 1991. Applications received prior to October 15, 1990 and March 1, 1991 will receive full consideration for January and September openings, respectively. Interested persons should send a resume and a list of references to Bill D. Carroll, Professor and Chairperson, Computer Science Engineering Department, P.O. Box 19015, The University of Texas at Arlington, Arlington, TX 76019. Phone: 817-273-3785. FAX 817-273-2548, Internet: carroll@evax.arl.utexas.edu.

The University of Texas at Arlington is an Equal Opportunity Affirmative Action Employer.

**ACADEMIA SINICA**
**Taiwan, Republic of China**
**Institute of Information Science**

Applications are invited for research position in Institute of Information Science, Academia Sinica. Ph.D. in Computer Science or closely related fields required. Demonstratable research ability necessary. Applicants for senior positions must have proven research record. All fields in Computer Science are welcome.

The Institute offers a good research environment. No duty of teaching. Facilities include a 32-node NCUBE 2 parallel supercomputer, many SUN, SGI, and E&S workstations. An easily accessible ETA-10Q supercomputer is in the Academia Sinica.

Interested people please send application to Dr. Y.S. Kuo, Acting Director, Institute of Information Science, Academia Sinica, Taipei, Taiwan, 11529, Republic of China. Fax: (001-886-2) 782-4814.

**UNIVERSITY OF SOUTHWESTERN LOUISIANA**
**The Center for Advanced Computer Studies**
**Faculty Positions**
**Graduate Fellowships**

Candidates with a strong research record and earned doctorate in Computer Science/Engineering are invited to apply for tenure-track and senior positions available starting in Fall 1991. Appointments are planned for Professor and Associate Professor ranks, with consideration given to exceptional candidates at the Assistant Professor rank. Candidates for the senior levels must have established publication and grant credentials. Consideration will be given to all outstanding applicants, but preferred areas of interest are software engineering, computer networks, operating systems, databases, computer architecture, artificial intelligence, and theoretical computer science.

Our typical teaching load is two graduate-level courses per year and a continuing research seminar. Substantial State and University funds are available to support research initiation efforts. Salaries are competitive and excellent support for travel, equipment, research assistants, and professional activities is provided so you can achieve your professional goals. Our Colloquium Series brings typically 8 world known professionals to our campus each year.

A number of Ph.D. Fellowships valued at up to $18,000 per year including tuition and fees are available. They provide support for up to 4 years of study towards the Ph.D. in Computer Science or Computer Engineering. Eligible candidates must be U.S. citizens or hold an earned MS degree from a U.S. or Canadian University. Recipients also receive preference for low-cost campus housing.

The Center is a graduate research center of 36 faculty and staff with programs leading to MS/Ph.D. degrees in Computer Science and Computer Engineering. The Center is located in Acadiana about 100 miles west of New Orleans. External grants/contracts support research in a wide range of areas. The Computing Research Laboratory includes a 60-node Sun network, an Encore parallel processing system, 2 VAX 11/780's, 2 Cogent XTM parallel computing systems, a comprehensive digital design lab, laser printers, plotters, FAX, and other equipment. Instruction utilizes a 3-processor Pyramid 90X network running UNIX and an IBM 3090-200 with a vector processor. Several other well-equipped laboratories suport research in Image Processing & Pattern Recogniton, VLSI Design, Parallel Computing and Graphical Information Systems, and Intelligent Robotic Machines. About 210 students are enrolled in computing graduate programs, including 100 for the Ph.D. The undergraduate program in the Computer Science Department is accredited by CSAB and offers both scientific and commercial options, with a current enrollment of 277. The undergraduate program in the Electrical and Computer Engineering Department is accredited by ABET and offers an option in Computer Engineering, with a current enrollment of 156.

To apply, send a copy of your resume and the names and addresses of at least three professional references. Applications will be considered until all positions are filled.

Dr. Michael C. Mulder, Director, The Center for Advanced Computer Studies, USL, Lafayette, LA 70504-4330. Phone: (318) 231-6284. E-Mail: cathy@cacs.usl.edu.

**UNIVERSITY OF NOTRE DAME**
**Chairperson**
**Department of Computer Science and Engineering**

The University of Notre Dame invites applications for the Chair of the recently established Department of Computer Science and Engineering in the College of Engineering. This position is a unique opportunity to provide significant input in establishing direction for the initiation and growth of programs in this department. Persons with established records in education and scholarship who would welcome the challenge of developing a new department and its programs are encouraged to apply. The faculty of the department is expected to determine the curricula leading to graduate and undergraduate degrees in computer science and in computer engineering.

The Department of Computer Science and Engineering has twelve tenure-track faculty positions. The Department of Electrical Engineering and its present graduate and undergraduate options of emphasis in computer engineering are expected to provide the foundation for the programs in the new department. It is anticipated that the Department of Computer Science and Engineering will have close cooperative ties with the Department of Electrical Engineering, Department of Mathematics, and other academic units.

Qualified applicants are encouraged to submit a resume and letter of interest to:

Dr. Panos J. Antsaklis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

For additional information, please contact the Dean's office at (219) 239-5534.

The University of Notre Dame invites applications from all qualified persons without regard to sex, ethnic origin, religious preference or physical impairment.

**UNIVERSITY OF CENTRAL FLORIDA**
**Computer Engineering**

The University of Central Florida College of Engineering invites applicants for tenure-track Assistant Professor positions in its Department of Computer Engineering beginning with the Fall term of 1991. A Ph.D. in Computer Engineering or a related discipline or substantial completion of the degree by the closing date is required. All interest areas will be considered although current preference areas include Software Engineering, Computer Control Systems, Real-time simulation, and Computer Architecture. UCF is a member of the State University System of Florida and has a current enrollment in excess of 21,000 students. The College of Engineering has over 3,500 students at present. The University is developing a Research Park adjacent to its main campus to support high-technology government and industrial activity in the Central Florida area. Computer facilities for instruction and research include a Sun Workstation Network, an NCUBE Parallel Machine and numerous mini and micro computers.

Send Resumes and the names of three references postmarked by February 15, 1991, to:

Dr. C.S. Bauer, Chair
Department of Computer Engineering
University of Central Florida
Orlando, FL 32816
Phone: (407) 823-2236,
FAX 407-823-5483

The University of Central Florida is an Equal Opportunity/Affirmative Action employer. As an agency of the State of Florida, the University makes all application materials and selection procedures available for public review.

**UNIVERSITY OF WISCONSIN-MILWAUKEE**
**Faculty Positions in Computer Science**

The Department of Electrical Engineering and Computer Science at the University of Wisconsin-Milwaukee is recruiting Computer Science faculty at the junior and senior levels. All candidates should have a commitment to research and teaching. Senior candidates are expected to have excellent research records. Areas of interest are in Artificial Intelligence, Software Engineering, Computer Networks, Parallel and Distributed Computation.

The Department offers undergraduate and graduate programs in Computer Science and has 12 full time Computer Science faculty members. Our current research strengths include Data Security, Cryptography, Fault Tolerant Computing, Computational Geometry, and Parallel and Distributed Computation. Computer Science research and instruction are supported by a modern computing environment. The University is located near the shores of Lake Michigan close to pleasant residential neighborhoods and lovely parks. Interested individuals are requested to send a resume to Professor K. Vairavan, CoChair for Computer Science, Department of Electrical Engineering and Computer Science, University of Wisconsin-Milwaukee, Milwaukee, WI 53201. The University of Wisconsin is an Affirmative Action/Equal Opportunity Employer.

## UNIVERSITY OF CALIFORNIA, DAVIS
**Faculty Positions in Electrical Engineering and Computer Science**

The Department of Electrical Engineering and Computer Science at UC Davis invites applications for various tenure track positions. The primary areas of interest are Computer Engineering and Microprocessor Application; Electronic Circuits; Image Processing and Computer Vision; and Optoelectronics. One position in the area of image processing and one in the area of optoelectronics is open to all ranks. Ohter positions are at the assistant professor level.

The department, with 53 faculty members and 180 full-time graduate students, is experiencing rapid growth. Our College is the nation's sixteenth largest producer of engineering Ph.D.'s in a University which has the nineteenth largest extramural research funding. Salary and benefits are extremely attractive.

Davis is a pleasant, family-oriented community near Sacramento, within easy driving distance to Silicon Valley, the Lawrence Livermore National Laboratory, San Francisco, the Pacific Ocean, and the Sierra Nevada Mountains.

We are seeking individuals with strong records of teaching and research and with ambitious plans. Senior appointments require outstanding records of achievement; junior appointments must show evidence of great promise. All faculty are expected to have a strong commitment to teaching at all degree levels, and to demonstrate the ability to attract significant research support.

The positions require a Ph.D. degree or equivalent, and are open until filled; but in order to assure consideration, applications should be received by March 1, 1991. Send a resume and the names of at least three references to:

Professor S. Louis Hakimi, Chair
Attention: Faculty Search Committee
Department of Electrical Engineering and Computer Science
University of California
Davis, CA 95616

The University of California, Davis, is an equal opportunity/affirmative action employer.

## CASE INSTITUTE OF TECHNOLOGY CASE WESTERN RESERVE UNIVERSITY

We invite applications for tenure track faculty positions at all levels. Candidates from all research areas will be considered, but the thrust research areas in the Department are VLSI systems and design automation, applied artificial intelligence and logic programming, database design and systems, and software systems and design environments. Candidates should have a Ph.D. in computer science or computer engineering or closely allied fields; competitive salaries will be offered to attract the best candidates.

CWRU is a private university with a total enrollment of 8,400, of which 5,100 are graduate and professional students. The Engineering School of Case Institute of Technology is among the top ten engineering schools in terms of research funding per faculty member and undergraduate student quality. The University campus is the hub of the pleasant area known as University Circle, an incorporation with neighboring cultural centers and museums, about five miles from downtown Cleveland.

The Department of Computer Engineering and Science has 14 faculty positions, and a graduate student body of 110 students, 40 of which are in the Ph.D. program. Departmental facilities are based upon an ethernet local area network, connected to INTERNET, which supports a UNIX operating system and about 40 SUN and other workstations. In addition, faculty and students participating in the Center for Automation and Intelligent Systems Research have access to the Center's computing facilities.

The Department recently acquired the Nord Professorship, supported by a donation of over one and a half million dollars, for which we invite distinguished senior faculty applicants. This position will provide additional funds for travel, graduate student support and equipment.

Applicants should submit their curriculum vitae and names of at least three references to: Lee J. White, Chairman, **Department of Computer Engineering and Science,** Case Western Reserve University, Cleveland, Ohio 44106; INTERNET: leew@alpha.ces.cwru.edu; candidates with previous academic experience may wish to provide at most three reprints of their most important publications.

An equal employment and affirmative action employer.

## UNIVERSITY OF SOUTHERN CALIFORNIA
**Computer Science**

Applications are invited for senior or junior tenure-track faculty positions in Computer Science. Candidates should have a strong commitment to research. Openings exist in most areas of experimental and theoretical computer science; with emphasis on operating systems, networks, compilers, artificial intelligence, graphics, geometric modeling, software engineering, and parallel algorithms and languages. The department currently has 19 full-time faculty, 135 doctoral students and annual research support excess of $3 million.

Research computing facilities include some 50 Sun 3 and Sun 4 workstations, a 32-node Intel IPSC/2-d5 hypercube, 4 TI Explorer Lisp machines, and numerous personal computers. An 8-processor Alliant and a 64K processor Connection Machine (located at the USC Information Sciences Institute) are also available for research via high-bandwidth networks. Research laboratories are dedicated to Brain Modeling, Artificial Intelligence, Programmable Automation, Robotics, Databases, Networking, Graphics and Animation, and Software Engineering. Teaching is supported by an additional 100 Sun workstations. Interested candidates should send a resume and a list of references to: Prof. Ellis Horowitz, Acting Chairman, Computer Science Department, SAL 200, University of Southern California, Los Angeles, CA 90089-0782.

USC is an Equal Opportunity/Affirmative Action Employer.

## BALL STATE UNIVERSITY MUNCIE, INDIANA
**Announcement of Position Vacancy Computer Science**

Faculty position available Fall 1991 in any major area of computer science. Background in software engineering, operating systems, communication, or compilers preferred. Faculty will teach computer science courses for masters degree and undergraduate students. An active scholarship program is expected. Minimum Qualifications: Ph.D. in computer science; ABD acceptable if degree is completed by Fall 1992. Preferred Qualifications: Ph.D. in computer science; at least two years experience in software development and teaching; excellent teaching record and several publications. Send resume, three letters of reference and official transcripts to Dr. Wayne M. Zage, Chairperson of Search and Selection Committee, Department of Computer Science, Ball State University, Muncie IN 47306. Review of applications will begin February 15, 1991, and continue until the position is filled.

Ball State University Practices Equal Opportunity in Education and Employment.

## EAST STROUDSBURG UNIVERSITY

Ph.D. in Computer Science full-time continuing position available. The position will consist of teaching courses at both the graduate and undergraduate levels and directing master's research projects and theses. The department has well respected undergraduate and graduate programs and maintains a high quality of grant supported investigative research. Desirable geographic location. Rank and salary depending on background and experience. Excellent fringe benefits. Apply by Janaury 15, 1991. Affirmative Action/Equal Opportunity employer. Women and minorities especially urged to apply. Send resume, transcripts and names and addresses of three references to

Professor Richard G. Prince
Department of Computer Science
East Stroudsburg University
East Stroudsburg, PA 18301

## COMPUTER APPLICATIONS ENGINEER

Design and develop software for scientific and engineering applications. Derive requirements from high level system and interface definitions or operations scenarios and user descriptions. Prepare and complete high level and detailed designs and translate design into high level codes and assembly languages. Correct self-caused errors, develop test requirements, evaluate results, prepare reports, convey results to customers and staff. Requires MS computer science, 1 yr. experience in independent technical research, preparation of engineering reports, and presentation of results. Excellent communication skills required to advise and assist employees and customers. 40 hrs. wk, $2,700 per month. Reply Huntington Job Service, 914 Fifth Avenue, Huntington, WV 25713, (304) 538-5525, Job Order #WV0410726 and Job Order #WV0410614.

## MEMORIAL UNIVERSITY OF NEWFOUNDLAND
**St. John's, Newfoundland, Canada**
**Head, Department of Computer Science**

Applications and nominations are invited for the position of Head of the Department of Computer Science. The department offers undergraduate and graduate programs to the M.Sc. level, and is about to apply for a Ph.D. program. It has 23 faculty and 14 staff members. Active research areas include design and analysis of algorithms, parallel and distributed computations, software aspects of VLSI design, computer graphics, database concurrency control and recovery, petri net theory and applications, numerical analysis, software engineering and pattern recognition.

The department provides a UNIX software development environment supported by a network of RISC based workstations and servers. An optical fibre link allows access to additional university computing facilities such as VAX systems running VMS and a Convex system. An off-campus Amdahl 5890 is also available. Other machines are accessible through CA*NET.

Memorial University has a student population of about 15,000, and is the only university in the province of Newfoundland and Labrador. St. John's is the provincial capital with a population of over 150,000, and the oldest city in North America. It enjoys a moderate climate and offers numerous outdoor activities throughout the year.

Applicants should have demonstrated excellence in research and teaching in Computer Science, and be able to show leadership and administrative ability appropriate to the post. The appointment will normally be made at the rank of Professor. The deadline for receipt of applications is 1 January 1991.

Please address inquiries, nominations, and applications, including a full *Curriculum Vitae* and the names of at least three referees, to

Dr. Bruce Shawyer, Chairman,
Search Committee for Head of Computer Science,
Department of Mathematics and Statistics,
Memorial University,
St. John's, Newfoundland, Canada.
A1C 5S7.
Telephone: (709) 737-8783. FAX: (709) 737-3010.

Memorial University encourages both men and women to apply for this position.

In accordance with Canadian Immigration requirements, priority will be given to Canadian citizens and permanent residents of Canada.

## GEORGE MASON UNIVERSITY
**School of Information Technology and Engineering**
**Chair, Department of Computer Science**

Chair, Department of Computer Science, George Mason University. The School of Information Technology and Engineering at George Mason University invites nominations/applications for the position of Chair of the Department of Computer Science. The School seeks an individual with strong administrative qualities who will provide innovative and energetic leadership. The successful candidate must possess credentials of the highest quality, including an earned doctorate and an established reputation in Computer Science. George Mason University (GMU) is a state university located in Fairfax County, Virginia, which is a heavily concentrated area of computer-oriented high technology industries. GMU is just 16 miles from the cultural and cosmopolitan activities of Washington, D.C. The Department of Computer Science (CS) currently has 20 full-time faculty (plus a number of visitors and adjunct faculty). We offer BS and MS degrees in Computer Science, and a Ph.D. in Information Technology with a specialization in computer science. In addition to the Department of CS, the School of Information Technology and Engineering has Departments of Electrical and Computer Engineering, Information Systems and Systems Engineering, and Operations Research and Applied Statistics. The Department of CS has special research laboratories in artificial intelligence, parallel computation, image processing/vision, neural networks, and software engineering. Specialized computer laboratories of other departments are also available. The Department has Microvax, Suns, and Mac workstations, transputer systems, hypercubes, and other computing facilities. There are approximately 500 undergraduate BSCS majors, 260 graduate MSCS majors, and 360 doctoral students in the interdisciplinary Ph.D. program. The new Chair is expected to provide leadership in identifying potential areas of research, recruiting new faculty, enhancing research funding, and developing new programs in the Department. Inquiries from candidates for this position should be sent to Professor Carl Harris, Chair, CS Chair Search Committee, Room 201, Science and Technology I, George Mason University, Fairfax, Virginia 22030. Documents may also be sent electronically to charris@gmuvax.gmu.edu (INTERNET) or charris@gmuvax.bitnet. GMU is an Equal Opportunity/Affirmative Action Employer. Closing Date is March 1, 1991.

## THE UNIVERSITY OF ALABAMA

The University of Alabama, Tuscaloosa, invites applications for the position of department head in Computer Science. The department head reports to the Dean of the College of Engineering and has both teaching and administrative responsibilities. Applicants must hold the Ph.D. in Computer Science or a closely related field and have demonstrated therein, a strong commitment to teaching and research. The new department head must be able to provide leadership in increasing sponsored research and building the Ph.D. program. Experience as an administrator in an academic setting is desired. Rank and salary will be commensurate with qualifications. Please submit a resume and names, addresses, and telephone numbers of three references to: Dr. William G. Nichols, Box 870290, Tuscaloosa, AL 35487-0290. The search committee will begin its review process January 7, 1991, but applications will be accepted until the position is filled. The University of Alabama is an affirmative action, equal opportunity employer.

## RUTGERS, THE STATE UNIVERSITY OF NEW JERSEY
**Federal Aviation Administration Fellowships**

For the M.S. and Ph.D. degrees. Research in all areas related to aviation safety. Applications due by February 15, 1991. Write Graduate Director, Electrical Engineering, Piscataway, NJ, 08855-0909, or write Graduate Director, Computer Science, New Brunswick, NJ 08903.

## UNIVERSITY OF SOUTHERN CALIFORNIA

The Computer Engineering Division of the Department of Electrical Engineering-Systems at the University of Southern California is expanding, and looking to fill positions at the Assistant, Associate, and Full Professor level in the following areas: VLSI/CAD, networks (optical type), and architecture with an emphasis on hardware. Additionally, we are looking for a full-time instructor (M.A. only required) to support the Computer Science/Computer Engineering undergraduate degree program. For all openings, please send a resume and the names of at least three academic references to Jerry M. Mendel, Chairman, Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA 90089-0781. USC is an equal opportunity/affirmative action employer.

## WASHINGTON UNIVERSITY

Washington University in St. Louis seeks qualified candidates for the position of Professor and Chair of the Department of Computer Science, with a desired starting date of July 1, 1991. We are interested in candidates with a strong research record, with a dedication to excellence in undergraduate and graduate education and with a demonstrated potential for administration and leadership.

The Department has an excellent undergraduate program as well as a strong and expanding graduate program. The primary research concentrations are in distributed systems, advanced communication networks and intelligent computer systems with an emphasis on visualization as a tool in each case. The Department plans to continue building on these areas of strength as well as expanding into new areas. There are 15 regular faculty in the Department and 85 graduate students, as well as an excellent technical support staff and a large pool of affiliate faculty. Departmental laboratory facilities are very good and include a visualization laboratory, a systems prototyping lab, an NCUBE parallel computer, a variety of compute servers and ubiquitous workstations.

Washington University has a longstanding commitment to the principle that all candidates should be afforded equal opportunity regardless of age, race, sex or physical disability. Candidates must send a curriculum vitae and a list of references to: Professor C.I. Byrnes, Search Committee for the Computer Science Chair, Campus Box 1040, Washington University, One Brookings Drive, St. Louis, MO 63130.

## ELECTRONICS DESIGN ENGINEER

Acting as project leader, develop and implement revolutionary system capable of controlling multiple access points (passcard readers), multiple users (cashiers and managers) and multiple Programmable Logic Controllers for access and revenue control, and security/theft prevention at 2,000 parking lots and garages situated in 80 cities. Design and develop electrical circuits, electronic components, visual imaging systems, real-world signal systems and multi-user networks; design microcomputer hardware and software; design small controllers and embedded controllers; design and build interface and switches for an extensive point-of-sale computer application systems. Develop license plate Optical Character Recognition systems to read and transmit license plate to host computer. Develop specialized local area networks to network multiple cashier and front desk terminals. Apply electronic engineering principles, research data and propose product specifications. Direct or coordinate manufacturing or building of prototype system. Plan and develop test program. $31,500 per annum, 40 hour work week. Must have B.S. in Electrical Engineering. Require one graduate level course in Very Large Scale Integrated Circuit Design. Inquire at Texas Employment Commission, Houston, Texas, or send resume to the Texas Employment Commission, TEC Building, Austin, Texas 78778, J.O. #5424763. Ad paid by an equal employment opportunity employer.

## SOUTHERN ILLINOIS UNIVERSITY AT CARBONDALE

Applications are invited for one or two tenure track faculty positions in Computer Science. One position is at the Assistant Professor level beginning August 16, 1991. A second, similar position opening, may become available depending upon funding. Candidates should have a broad competence in computer science and/or computer engineering. All fields of specialization will be considered, with the Department seeking to strengthen the areas of operating systems, software engineering and networks. Evidence of on-going and future research, a commitment to teaching and willingness to participate fully in the Department's graduate and undergraduate programs are basic requirements. Applicants should have, or expect to receive in 1991, a Ph.D. in Computer Science or Computer Engineering. Departmental facilities include a network of personal computers, Sun and VAX workstations and a shared-memory parallel machine—the Sequent Balance 8000. IBM mainframe computers, including a 3090 with vector processing and various network capabilities, are also available for research and teaching.

Applications will be accepted until March 1, 1991, or until the positions are filled. Resumes and three letters of reference should be sent to: Faculty Recruitment Committee, Department of Computer Science, Southern Illinois University at Carbondale, Carbondale, IL 62901-4511.

SIUC is an Equal Opportunity, Affirmative Action Employer.

## AUBURN UNIVERSITY
**Earle C. Williams Eminent Scholar Chair in Electrical Engineering**

Nominations and applications are invited for the Earle C. Williams Eminent Scholar Chair in Electrical Engineering. Candidates for this chair should have achieved national and international prominence in digital systems and/or microelectronics.

Applicants or nominees must have an earned doctorate, senior academic experience, and a documented record of distinction in university teaching and research. The successful candidate will be expected to provide intellectual leadership in his/her area of expertise for the Department of Electrical Engineering as well as enrich the scholarly environment at Auburn University.

Auburn University is located in the city of Auburn in east-central Alabama. This land-grant university enrolls more than 21,000 students, the largest on-campus enrollment in the state. The Department of Electrical Engineering, one of eight departments within the College of Engineering, offers Bachelor, Master, Master of Science and Ph.D. degrees in Electrical Engineering. The department has a current enrollment of 939 undergraduate students and 100 graduate students. The 28 full-time faculty have an annual research expenditure of approximately $2 million.

The Search Committee will begin its review of applications immediately. Interested candidates should submit: (1) a detailed resume, (2) a letter indicating an interest in the chair, the candidate's academic philosophy, and a brief statement of accomplishments in teaching and research, and (3) names and addresses of five references. Nominations should be submitted with the complete name, mailing address and telephone number of the individual nominated.

Applications and nominations should be sent to Professor J. David Irwin, Department of Electrical Engineering, Auburn University, AL 36849-5201. Auburn University is an affirmative action/equal opportunity employer. Applications from minority and female candidates are encouraged.

## CASE INSTITUTE OF TECHNOLOGY
**NORD Professorship in Computer Engineering and Science**

The Department of Computer Engineering and Science at Case Institute of Technology is seeking a nationally recognized scholar and researcher to fill the NORD Professorship. This position was recently established by the donation of over one and a half million dollars, which will provide outstanding professional opportunities and a highly competitive salary, together with additional funds for travel, graduate student support and equipment. The qualifications include a Ph.D. in computer science, computer engineering or closely allied fields, and an ability to establish and develop external support for a nationally recognized research program in computer science/computer engineering. We invite applications from senior faculty at both the associate professor and full professor levels.

CWRU is a private university with a total enrollment of 8,400, of which 5,100 are graduate and professional students. The Engineering School of Case Institute of Technology is among the top ten engineering schools in terms of research funding per faculty member and undergraduate student quality. The University campus is the hub of the pleasant area known as University Circle, an incorporation with neighboring cultural centers and museums, about five miles from downtown Cleveland.

The Department of Computer Engineering and Science has 14 faculty positions, and a graduate student body of 110 students, 40 of which are in the Ph.D program. Departmental facilities are based upon a ethernet local area network, connected to INTERNET, which supports a UNIX operating system and about 40 SUN and other workstations. In addition, faculty and students participating in the Center for Automation and Intelligent Systems Research have access to the Center's computing facilities.

Applicants should submit their curriculum vitae and names of at least five references to: Lee J. White, Chairman, **Department of Computer Engineering and Science,** Case Western Reserve University, Cleveland, Ohio 44106; INTERNET: leew@alpha.ces.cwru.edu; applicants may wish to provide at most three reprints of their most important publications.

An equal employment and affirmative action employer.

## WRIGHT STATE UNIVERSITY
**Department of Computer Science and Engineering**
**Dayton, Ohio 45435**

Applications are invited for the position of INSTRUCTOR in the Department of Computer Science and Engineering to teach undergraduate courses. Instructor is a non-tenure track position. At least an MS in CS, CEG, or a related discipline is required with some teaching experience preferred. Review for positions will begin December 1, 1990 and continue until filled. Send applications, resume, three references, and official transcripts to Dr. A.D. McAulay, Department of CS&E, Wright State University, Dayton, OH 45435. Wright State University is an EO/AA employer.

## TRANSYLVANIA UNIVERSITY
**Computer Science**
**Assistant/Associate Professor**

Full-time, tenure track, Ph.D. in computer science or master's in computer science with Ph.D. in a closely related field. Rank and salary dependent upon background. Exceptionally well qualified candidates may be considered for a Bingham Award for Excellence in Teaching; smaller, "start-up" grants are available for less experienced faculty. This recognition provides a supplement of up to 50% of base salary for the position. Transylvania is a private, liberal arts college with a strong commitment to excellence in undergraduate education. The program in computer science is of long standing and is recognized for its outstanding quality. Please send letter of application, curriculum vitae, and names of three references to Dr. Dwight W. Carpenter, Computer Science Program, Transylvania University, Lexington, KY 40508. An Equal Opportunity Employer.

## THE UNIVERSITY OF TAMPA
**Computer Information Systems**

The College of Business, CIS Department, announces a CIS position with teaching responsibilities in all areas of the curriculum, commencing August 1991. A Ph.D. in CIS/MIS/CS is preferred although ABD's acceptable. Knowledge of Artificial Intelligence, Object Oriented Design, Computer Graphics, and/or Decision Support Systems is desired. The College of Business rewards teaching excellence and encourages faculty contact with the business community. This position is a tenure-track with rank and salary dependent on qualifications.

Applicants should send a letter of application and an attached vita by January 1, 1991 to:

Dr. Gordon Couturier, Coordinator
College of Business, CIS Department
The University of Tampa
401 West Kennedy Boulevard, Box 131F
Tampa, FL 33606-1490

The University of Tampa is an Equal Opportunity, Affirmative Action Employer and encourages women and minorities to apply.

## THE UNIVERSITY OF GEORGIA
**Associate Vice President for Computing and Networking Services**

Applications and nominations are invited for the Associate Vice President for Computing and Networking Services at the University of Georgia. The University of Georgia, located in Athens, enrolls 28,000 students in thirteen colleges and schools. It is the flagship university among the 34 state-supported institutions of higher education comprising the University System of Georgia.

The Associate Vice President for Computing and Networking reports to the Vice President for Academic Affairs and is responsible for campus-wide leadership, strategic planning, and coordination of the computing and networking facilities required by a comprehensive, modern university. S/he serves as a focal point for the articulation and implementation of an institutional vision of the optimal role of computing and networking in support of the University's teaching, research and service activities. In pursuit of these responsibilities the Associate Vice President will have oversight responsibility for the central computing facilities and campus data communications network services. S/he will interact with the instructional, research, library and administrative communities.

Qualified applicants should:

- be familiar with the diversity of computing activities ongoing at a major research university,
- be familiar with telecommunications networks and distributed computing,
- have a strong technical background which includes significant computing experience and broad acquaintance with computing applications,
- have excellent oral and written communication skills,
- have strong leadership skills which are aligned with state-of-the-art, successful academic computing environments,
- possess a degree from a recognized institution of higher learning.

It would be advantageous if applicants have credentials to qualify for a faculty appointment.

Applications and nominations postmarked by January 15, 1991, are promised full consideration by the Search Committee. Complete applications must include a current resume plus names and addresses of three references. Candidates are assured of maximum confidentiality permitted by state law, and no reference will be contacted until the candidate is first notified. The position becomes available beginning July 1, 1991. Please send applications and nominations to:

Dean John Kozak
Chairman, Screening Committee
The University of Georgia
Franklin College of Arts and Sciences
Athens, GA 30602

## THE GEORGE WASHINGTON UNIVERSITY
**School of Engineering and Applied Science**
**Visiting Professorships**
**Research Faculty/Research Staff**

Visiting Professorships, Research Faculty, and Research Staff Positions, at junior and senior levels, are available in the School of Engineering and Applied Science, The George Washington University starting Fall Semester 1991. The School of Engineering and Applied Science is organized into four academic departments: the Department of Civil, Mechanical and Environmental Engineering; the Department of Electrical Engineering and Computer Science; the Department of Engineering Management; and the Department of Operations Research.

Candidates are especially sought to teach and/or conduct research in the following areas: Aeronautics; Aerospace Engineering; Analog Electronics/VLSI; Astronautics; Biotechnology Management; Communications; Computational Fluid Dynamics; Computer Aided Design; Computer Engineering; Computer Graphics; Computer-Integrated Design and Manufacturing; Computer Science; Decision Analysis; Decision Support Systems; Electrical Engineering; Engineering Management; Environmental Management; Environmental Engineering; Finite Element and Mechanics; Geotechnical Engineering; Information Technology Management; Manufacturing/Production Management; Mathematical Optimization; Operations Research; Project and Program Management and Total Quality Management; Reliability; Robotics/Controls; Simulation; Software Systems Engineering; Stochastic Processes; Structural Engineering; Technology Assessment and Transfer; and User-Computer Interface.

Appointments are for one-year periods. Applicants should send vita, including complete publication list, and three references to:

Visiting Engineers Scholors Program
OR
Research Faculty and Staff Program
School of Engineering and Applied Science
The Geoerge Washington University
Washington, D.C. 20052

The George Washington University is an Affirmative Action/Equal Opportunity Employer.

## THE INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING

(ICASE) is seeking postdoctoral fellows or visiting senior researchers. Active research areas in computer science at ICASE include design and implementation of tools and compilers for distributed memory and SIMD multiprocessors, performance modeling and prediction of miltiprocessor algorithms and architectures, analysis of shared virtual memory mechanisms, and parallel algorithms for sparse matrix problems and for solving partial differential equations via adaptive and unstructured mesh methods.

ICASE has access to a variety of multiprocessor computers both locally and via high bandwidth networks.

ICASE has an iPSC/860 along with local access to a Cray-2 and a Cray Y-MP. We also have a high bandwidth link to the NASA Ames Research Center where access to a CM-2 and other machines may be arranged. ICASE operates its own network with the usual assortment of Suns and graphics workstations.

We have close academic affiliations with a wide range of universities and institutes and maintain a very active summer visitor program.

Applicants should respond by e-mail to rgv@icase.edu or should send resumes and descriptions of proposed research to:

Dr. Robert G. Voigt or Joel Saltz
Director, Lead Computer Scientist
ICASE
Mail Stop 132C
NASA Langley Research Center
Hampton, VA 23665

ICASE is an Equal Opportunity Employer.

## UNIVERSITY OF CALIFORNIA, SANTA CRUZ

Computer Engineering Department invites faculty applications for two positions.

One Associate or Full Professor, and one Assistant Professor with an application closing date of January 1, 1991. Preference will be given to candidates in the following areas:

One with research and teaching interests in graphics and imaging, with emphasis on applications and systems. An interest in workstation interfaces and networking is desirable. (Provision #233-901).

One with research and teaching concentration in computer systems architecture. A strong interest in hardware and design that would exploit our department's strengths in VLSI/CAD is desirable. (Provision #175-890).

Salary range for the tenured position, $57,000-$80,700 and for the tenure-track position, $46,800-$49,200 (9 month basis).

A Ph.D. in Computer Engineering, Electrical Engineering, Computer Science or equivalent is required. For complete information please contact: Chair, Computer Engineering Faculty Search Committee, Baskin Center for Computer Engineering & Information Sciences, Applied Sciences Building, University of California, Santa Cruz, CA 95064. (Questions may be sent via email to recruit@saturn.ucsc.edn or recruit@ucsccrls.bitnet.) UCSC is an EEO/AA/IRCA employer.

## NJIT
## Faculty:
## Computer and Information Science

NJIT seeks assistant, associate and full professors for Spring/Fall 1991 in distributed computing including computer architecture, operating systems, data communications and networking, realtime computing and fault tolerance, software development including compiling, computer graphics, office automation, data management systems, information management systems, cognitive science, and computational linguistics; computer graphics and computer visions and other areas. Qualifications: Ph.D. in computer science or closely related field required; senior level applicants must have proven research and funding record.

The department offers B.S., B.A., M.S. and Ph.D. in computer science. Computing facilities include the $30 million Information Technologies Building with VAX 6400, VAX 8530, IBM 4361, SUN workstations, Symbolics machines, TI Explorers and graphic systems. Send resume and names of three references to: Personnel Box CIS.

NJIT is the technological university of New Jersey with nearly 8,000 students enrolled in Newark College of Engineering, the School of Architecture, the College of Science and Liberal Arts and the School of Industrial Management.

NJIT does not discriminate on the basis of sex, race, color, handicap, religion, national or ethnic origin or age in employment.

NEW JERSEY INSTITUTE OF TECHNOLOGY
University Heights
Newark, NJ 07102

## DARTMOUTH COLLEGE
## Computer Science

Dartmouth College invites applications for positions in computer science and engineering. This solicitation represents a joint recruiting effort of the Department of Mathematics and Computer Science and the Thayer School of Engineering for faculty who will serve as faculty in the Ph.D. Program in Computer Science as well as hold an appointment in either the Department of Mathematics and Computer Science or the Thayer School of Engineering. Faculty in these positions teach at the graduate level and conduct research under the auspices of the Ph.D. Program and also teach undergraduates in their respective units. Candidates must excel in both teaching and research. A Ph.D. in computer science, computer engineering, or a related field is required.

Program faculty have Sun, IBM, and DEC workstations in their offices with network connections to a number of VAX, IBM, and Honeywell mainframes, as well as Convex and Alliant minisupercomputers. Microprocessor development and CAD systems, graphics terminals, and microprocessor laboratories are also available.

**Department of Mathematics and Computer Science**

Applications are invited for tenure track positions in Computer Science at all levels, Assistant, Associate, and Full Professor. Candidates in languages, systems, and artificial intelligence are especially encouraged to apply. There are currently ten Computer Science faculty in the department, which conducts the undergraduate major in Computer Science at Dartmouth. Current research includes algorithm analysis and design, computer languages and systems, theory, computational geometry, databases, parallel and distributed computation, computer vision, system security, logic programming, and signal processing.

Interested persons should submit a resume and names of three references to Prof. Donald B. Johnson, Department of Mathematics and Computer Science, Bradley Hall, Dartmouth College, Hanover, NH 03755. Review of applications will begin in January, 1991, and will continue until the search is complete.

**Thayer School of Engineering**

Applications are invited for senior and junior tenure track appointments. Significant expansion is in progress with additional new positions anticipated during the next few years. Of special interest are candidates in VLSI, with CAD experience and an interest in system design. Current research in computer engineering has focused on the design of special purpose computational structures, with the intent of supporting areas of scientific research that can benefit from customized computing power. A Rapid Prototyping Laboratory is being developed for the construction of these digital systems, so candidates with an interest in developing prototype systems are particularly encouraged to apply.

Interested persons should submit a resume and names of three references to Prof. Barry S. Fagin, Thayer School of Engineering, Dartmouth College, Hanover, NH 03755. Review of applications will begin in January, 1991, and continue until the positions are filled.

Dartmouth College is an equal opportunity/Affirmative Action employer and encourages applications from women and members of minority groups.

## EASTERN CONNECTICUT STATE UNIVERSITY

The Mathematics/Computer Science Department of Eastern Connecticut State University invites applications for two tenure-track, rank-open appointments beginning September 1991. Candidates must have a Ph.D. in Computer Engineering or Computer Science. Applicants must have a strong commitment to undergraduate teaching, a willingness to write grants, the ability to conduct research in computer science or mathematics, and an interest in participating in the ongoing development of a liberal arts Computer Science program based on ACM guidelines. Screening of applicants will begin immediately and continue until the position is filled. ECSU is an Affirmative Action/Equal Opportunity Employer. Send letter of interest, a current vita and arrange for three letters of recommendation to be sent to: Dr. C. Gary Rommel, Chair, Department of Mathematics/Computer Science, Media 251, Eastern Connecticut State University Willimantic, CT 06226.

## SOFTWARE SUPPORT ENGINEER

Software Support Engineer needed to conduct high level system design and implementation of feature development. Research and resolve software problems encountered during the product/feature verification process. Provide a centralized support of the on-site verification personnel for the customer. Provide design and customer support for digital telecommunications switching products. Provide technical support as required by design to reproduce, clarify, and resolve customer problems, including testing in the captive office and on site. Requires a Bachelors' Degree in Computer Science or its' equivalent and 1 year experience in job offered, or 1 year directly related telecommunications Software experience. Or will consider a Masters' Degree in Computer Science in lieu of Bachelors' Degree and experience. Background should include 2 semesters or 6 months experience in network systems, and computer architecture (which includes computer logic design). 40 hour work week. $42,200 per year. Apply at the Texas Employment Commission, Dallas, Texas, or send resume to the Texas Employment Commission, TEC Building, Austin, Texas 78778 Job Order #5515177. Ad Paid By An Equal Employment Opportunity Employer.

## POLYTECHNIC UNIVERSITY
## Computer Science Faculty Positions

The Department of Computer Science invites applications for faculty members at all ranks. Candidates must have exceptionally strong research and publication records. We are especially interested in candidates with a major interest in databases, and network analysis and design, however, candidates with interests in other areas are also invited to apply. A Ph.D. in computer science or computer engineering is required.

Polytechnic has three campuses within the Metropolitan New York area and one of the largest graduate computer science programs in the United States, at both the Masters and Ph.D. levels. The New York area offers substantial opportunities for research interaction with industry.

The Department of Computer Science is part of the School of Electrical Engineering and Computer Science. Major research interests within our faculty include algorithms, computer architecture distributed systems, programming languages, computer vision and image understanding, and software engineering. Several faculty members are actively engaged in research with our Center for Advanced Technology in Telecommunications in the areas of network design and management, and in the area of distributed systems.

Please send a resume and the names of at least three references to Prof. Gad Landau, CS Search Committee Chair, Department of Computer Science, Polytechnic University, 333 Jay St., Brooklyn, NY, 11201. Applicants must be U.S. citizens or have permanent resident status.

Polytechnic is an Equal Opportunity Employer, M/F/H/V.

## CLEMSON UNIVERSITY
**Department of Electrical and Computer Engineering**

The Department of Electrical and Computer Engineering at Clemson University invites applications for tenure/tenure-track positions, primarily at the assistant professor level. A Ph.D. is required. Candidates with research interests in one of the following areas are sought: communications and digital signal processing; networks and distributed computing; quantitative analysis of computer architectures; electromagnetic analysis of active devices (microwave, millimeter wave, or electro-optic); and multidisciplinary applications in power systems (e.g. power and artificial intelligence; power and computer communications). A successful candidate must exhibit exceptional potential for research and teaching.

Clemson University's College of Engineering was listed as one of the United States' "up-and-coming" engineering graduate programs in the March 19, 1990, issue of *U.S. News and World Report*. The ECE Department comprises thirty-six full-time faculty, approximately 700 undergraduate students, and 140 graduate students. It offers B.S., M.S., and Ph.D. degrees in both electrical engineering and computer engineering. Facilities and/or groups bearing on the areas indicated above include the Clemson University Electric Power Research Association (CUEPRA); a microcircuits reliability research group with a class 100 clean room; automated microwave measurement facilities to 26 GHz; an image processing laboratory; a Center for Computer Communications Systems.

Resumes, supported with a list of references, should be sent to L. Wilson Pearson, Head; Department of Electrical and Computer Engineering; 102 Riggs Hall; Clemson University; Clemson, SC 29634-0915. Initial screening of applicants will begin January 15, 1991 and continue until positions are filled. Clemson University is an Equal Opportunity/Affirmative Action Employer.

## THE VIRGINIA MILITARY INSTITUTE
**Mathematics/Computer Science**

A tenure-track position beginning August, 1991. Applicant should have a strong interest in teaching.

Preference will be given to applicant with a Ph.D. in computer science. Applicants with significant progress toward a Ph.D. will be considered. Duties include teaching both computer science and mathematics. Salary and rank commensurate with qualifications.

VMI is state-supported with 1300 undergraduates in engineering, liberal arts, and science. It is located in an attractive college town with three colleges within a six mile radius. Faculty wear uniforms but have no other military duties.

Deadline for applications is February 1, 1991, but will be extended as necessary. Send resume, three letters of recommendation, and a graduate transcript to George Piegari, Department of Mathematics and Computer Science, Virginia Military Institute, Lexington, VA 24450.

AA/EEO Employer.

## MASSACHUSETTS INSTITUTE OF TECHNOLOGY
**Faculty Positions**

The Department of Electrical Engineering and Computer Science seeks candidates for faculty positions starting in September 1991. We anticipate openings for several junior faculty appointments for individuals who are completing, or who have recently completed, a doctorate. Senior faculty positions may also be available in some areas. Faculty duties include teaching at both the graduate and undergraduate levels, research, and supervision of theses.

We are interested in candidates in most areas of electrical engineering and computer science, such as artificial intelligence, communications, computer systems and languages, flexible manufacturing, and solid-state materials and devices.

All candidates should write to the address below, describing their professional interests and goals. Applications should include a curriculum vitae and the names and addresses of three or more references. Additional material describing the applicant's work, such as papers or technical reports, would also be helpful. All candidates should indicate citizenship and, in the case of non-US citizens, describe their visa status.

Send all applications to:
Prof. F.C. Hennie
Room 38-435
Massachusetts Institute of Technology
Cambridge, MA 02139

M.I.T. is an equal opportunity/affirmative action employer.

## UNIVERSITY OF CALIFORNIA RIVERSIDE
**Department of Computer Science**

Applications are invited for tenured or tenure track positions in Computer Science beginning July 1, 1990 or later. Ph.D. and excellence in research and teaching are required. Two open rank positions are available. One position is targeted to Design Technology and one to Systems (architecture, software). Other areas may also be considered. The duties of the positions include teaching at both the undergraduate and graduate levels, research, and participation in the Department and the Computer Science program.

Prominent senior candidates are especially encouraged to apply. Candidates will have an opportunity to participate in shaping a new developing program in Computer Science, including creating and leading new research groups.

Send all materials including curriculum vitae and the names of at least three references to: Professor Marek Chrobak, Chairman, Computer Science Recruiting Committee, Department of Computer Science, University of California, Riverside, CA 92521. The pool of candidates will consist of all those whose completed applications are received by March 8, 1991. A complete application shall consist of the cover letter, the curriculum vitae, three letters of recommendation, and the publication list.

University of California, Riverside, is an Affirmative Action/Equal Opportunity Employer.

## THE UNIVERSITY OF WEST INDIES, ST. AUGUSTINE, TRINIDAD

Applications are invited for the following post:

Professor/Senior Lecturer in Mathematics. The Department wishes to strengthen its research interests in Discrete Mathematics including Combinatorics and Graph Theory and Applied Mathematics including Fluid Dynamics.

ANNUAL SALARY RANGE: Professor: TT$91,788-$111,372. Pension, Passages, housing, travel grant. Applications detailing qualifications and experience and naming three referees to the Registrar as soon as possible. Further particulars sent to all applicants.

## PROGRAMMER
**Engineering Computer Graphics**

The initial assignments for this job are designed to develop competence in applying programming procedures, coupled with computer graphics educational background, to routine problems. As a beginner programmer, the duties involve writing routine new programs using prescribed graphics specification in "C," Fortran, and Assembly under the DOS and UNIX operating environment. This job involves this person being directly supervised. The supervisor is the Vice President of Software development and inclusive of the above candidate has a total of four programmers reporting to him. This programmer will not have anyone reporting to him, Master's degree in Mechanical Engineering is required. Must have had at least two courses in computer graphics which deal with 2 and 3D primitives, interactive graphics and devices, GKS/PHIGS standards and engineering graphics application. And at least one course each in programming in C/UNIX and assembly. Also, must have worked on a major research project at the graduate level in the area of surface rendering and shading of graphics primitives. 40 hours per week, and $634.62 per week. Apply at the Texas Employment Commission, Houston, Texas, or send resumes to the Texas Employment Commission, TEC Building, Austin, Texas 78778, J.O. #5515180 and Ad Paid by an Equal Employment Opportunity Employer.

## YALE UNIVERSITY

The Department of Electrical Engineering, Yale University, invites applications for two senior faculty positions in computer engineering. The preferred area is computer architecture. Of particular interest are the design of high-speed computer systems, including multi-purpose parallel architectures, and the coordinated design of algorithms and architectures. Applicants should be recognized leaders in the field with demonstrated ability to initiate and lead a research program. Submit resumes ot Professor P.M. Schultheiss, Chairman, Department of Electrical Engineering, Yale University, P.O. Box 2157 Yale Station, New Haven, CT 06520. Yale University is an affirmative action, equal opportunity employer.

## UNIVERSITY OF BALTIMORE

Applications are invited for a tenure track position as Assistant Professor which will begin, Fall, 1991. Qualifications include a Ph.D. in Computer Science, potential for excellence in teaching undergraduate Computer Science and interest in research. The University has a large number of part-time students and offers many evening classes. Experience teaching in such an environment would be considered a plus. Candidates should have a letter of application, a curriculum vita, and name, address, phone number of at least three references sent to Search Committee, Department of Computer Science, Mathematics, and Statistics, University of Baltimore, 1420 N. Charles Street, Baltimore, Maryland 21201. Applications received by January 15, 1991 are insured full consideration. The University of Baltimore is an equal opportunity employer.

## SIMON FRASER UNIVERSITY
**Faculty Positions**
**School of Computing Science and Centre for Systems Science**

Applications are invited for tenure-track positions at all ranks, subject to budgetary authorization. Although, outstanding candidates in all areas of Computing Science will be considered, we are particularly interested in a person in systems.

A Ph.D. in Computing Science (or equivalent) is required and candidates should have a record of (or strong potential for) research and publications, graduate student supervision, and teaching.

The School of Computing Science, has 29 faculty members and offers M.Sc. and Ph.D. degrees in computing science as well as B.Sc. and B.A. degrees in computing science, B.Sc. honors degrees in computing science, digital systems (including VLSI) design, and math/computing.

The Centre for Systems Science is a multidisciplinary research organization which promotes excellence in technology-based areas such as intelligent systems, computer and communication systems, and microelectronics. Through the Centre, Fellowships from the B.C. Advanced Systems Institute are available to outstanding applicants, thereby making additional teaching release and infrastructure support possible.

Together, the CSS and the School have an impressive research network. The net consists mainly of SPARCStations and other SUN workstations with several LISP machines and high resolution colour workstations, and plotters for AI, graphics and VLSI design. All faculty offices are connected to the network. We are located in the new Applied Sciences building where we have a diverse collection of research laboratories. Teaching facilities include an instructional laboratory based upon SUN workstations running UNIX, and various microcomputer and hardware laboratories.

Simon Fraser University is situated on top of Burnaby mountain and serves about 16,000 students. Lying just east of Vancouver, the site commands magnificent views of Burrard Inlet, the North-Shore mountains, the Fraser River, and Vancouver harbour. The School also has links to a newly established downtown Vancouver campus. This lower mainland area of British Columbia is unique in Canada for its mild climate and varied recreational facilities.

Preference will be given to candidates who are eligible for employment in Canada at the time of application. Simon Fraser University is committed to the principle of equity in employment and offers equal employment opportunities to qualified applicants. Applications will be accepted until the positions are filled, although a practical cutoff date for 1991 is April 1st. To apply, send a curriculum vitae, evidence of research productivity (selected reprints), and the names, addresses, and phone numbers of three referees to:

Arthur L. Liestman, Director
School of Computing Science
Simon Fraser University
Burnaby, British Columbia, Canada
V5A 1S6
FAX: (604) 291-3045

## COLUMBIA UNIVERSITY
**Research Programmer**

Columbia University Computer Science Department seeks applicants at Staff Associate level. (BS required, MS preferred, minimum one year research experience) to participate in programming systems research under faculty supervision. Position involves prototype design, C/Unix programming, oral/written presentation, project management, supervision of students. Experience building software development environments, distributed object systems, AI applications desirable. Send resume to Ms. Ting Bell, Columbia Univ., Computer Science Dept., New York, NY 10027. Equal opportunity employer. We are interested in receiving applications from qualified women and minorities.

## HARVEY MUDD COLLEGE
**Senior Position in Computer Science**

Harvey Mudd College has re-opened its search for a senior professor of computer science to lead the department and help design a curriculum for a major in computer science. The successful candidate for this position will be able to provide leadership and direction both for the college's overall computer science program and for the new major. He or she must also desire the challenge and opportunity to develop a program which builds on the existing strengths of the college in science and engineering. Qualifications for the position include a doctorate in computer science or in a related field with computer science experience, demonstrated commitment to excellence in teaching, a willingness and ability to participate in curriculum development, and significant professional experience.

Harvey Mudd College, one of the nation's most selective undergraduate colleges of engineering and science, is a member of the Claremont Colleges. The college is an equal opportunity/affirmative action employer. Please send resume and names of four references to: Nathaniel Davis
Dean of Faculty
Harvey Mudd College
Claremont, CA 91711

## UNIVERSITY OF IDAHO
**Department of Computer Science**

The Department of Computer Science at the University of Idaho invites applications for a tenure-track faculty position at the assistant professor level, however, outstanding applicants at higher levels will also be considered. The Department of Computer Science is in the College of Engineering and one of the college's highest priorities is to build a strong research program in computer science. While specialization in computing architectures, networks, and software engineering are preferred, outstanding candidates in all areas are encouraged to apply.

Qualifications for this position include an earned Ph.D. in Computer Science or a closely related field, teaching and research ability, potential for establishing a strong research program, and US citizenship or lawfully authorized alien worker status. Successful candidates are expected to pursue an active research program, perform graduate and undergraduate teaching, and supervise graduate students.

The department has 13 tenure-track faculty, approximately 250 undergraduate majors, and 30 graduate students. BS and MS degrees and currently offered with plans underway for offering the Ph.D. degree in computer science. The department is a component of the NASA Microelectronics Research Center at the University of Idaho.

The Computer Science department has approximately 40 Unix work stations (HP, Apollo, and DEC) which are networked and have connections to Internet and Bitnet. Numerous other college and university work stations and computer laboratories are available for faculty and student use.

Applicants should submit a curriculum vitae and three letters of reference to John Dickinson, Search Committee Chair, Department of Computer Science, University of Idaho, Moscow, ID 83843 (email:search 91@ted.cs.uidaho.edu). Applications will be accepted until February 1, 1991 or until suitable candidates are selected. The University of Idaho is an EO/AA employer and educational institution and specifically invites applications from women and minorities.

## RSA DATA SECURITY INC.

RSA Data Security Inc. invites applications for a position in research and advanced development. RSA Data Security is a small company specializing in the integration of cryptographic technology into commercial systems. Applicants should have a Ph.D. in Computer Science, Mathematics, or a closely related field.

Responsibilities of the position include developing new crytographic protocols and algorithms: evaluating cryptographic systems; and maintaining an active research program.

Applicants should send a resume and the names of three references to:

RSA Data Security
10 Twin Dolphin Drive
Redwood City, CA 94065
Telephone: (415) 595-8782
Electronic mail: burt@rsa.com

An Equal Opportunity/Affirmative Action Employer.

**SOFTWARE ENGINEER**

Software engineer sought to design and maintain real time application softwares and firmwares for robotic metal and plastic marking machines, design new softwares and modify existing softwares and write programs utilizing MOTOROLA 6809 Assembly and C languages; to modify and utilize a variety of programs in D-Base III PLUS and similar programs; he/she will confer with other personnel to obtain data on limitations and capabilities of the machinery and systems and provide consulting services to customers. Salary: $29,000.00/yr. Send resume to J. Gaston, Mo. Div. of Employment Security, 505 Washington, St. Louis, Missouri 63101. Refer to Job Order Number 405289.

**MISSISSIPPI STATE UNIVERSITY**
**Electrical and Computer Engineering and Engineering Research Center (Search Extended)**

Applications are invited for tenure track positions in Electrical and Computer Engineering at Mississippi State University with a joint appointment in the NSF Engineering Research Center for Computational Field Simulation. Positions require the Ph.D. The technical areas of interest are in computer architecture with special emphasis in parallel architecture, parallel computing software, and rapid system prototyping including IC design. They require demonstrated ability with potential for attracting and conducting research. Salaries are competitive and commensurate with degree and experience. Responsibilities include both research and teaching involving graduate and/or undergraduate instruction.

The Department of Electrical and Computer Engineering offers ABET accredited undergraduate programs both in Electrical and Computer Engineering and graduate degrees through the doctorate. It has 32 faculty, 15 professional and administrative support staff members and an enrollment of approximately 780 undergraduate and 100 graduate students. The Department has an excellent record of scholarly achievement and has a major role in the MSU/NSF Engineering Research Center for Computational Field Simulation.

This ERC combines research in solution algorithms, grid generation, computer architecture and computer graphics synergistically to develop and apply high-performance computational simulation of field problems on complex configurations. The Center maintains a strong interaction with industry and also stresses a cross-disciplinary educational program in computational and computer engineering at the undergraduate, graduate, and professional levels. The research program is interdisciplinary, combining faculty, staff and students from engineering, mathematics, and computer science. The ERC will occupy a new 44,000 sq. ft. building (completed late 1990) in the Mississippi Research Technology Park adjacent to the University campus.

Mississippi State University is a comprehensive land grant institution and is among the top 100 research-funded institutions in the United States as defined by the National Science Foundation. The College of Engineering is one of nine colleges/schools in the University. The University has over 13,000 students and 850 faculty members.

The Department of Electrical and Computer Engineering is one of eight academic departments in the College of Engineering. Engineering with approximately 115 faculty and 2,200 undergraduate and 200 graduate students, has funded research expenditures which exceed $10.0 million in FY90. The new Mississippi Research and Technology Park with associated industry is contributing greatly to the research programs of the college.

Applications will be accepted until the positions are filled. Interested persons should submit a complete resume, including details of relevant interests, expertise, and experience, along with names and addresses of at least three references to:

Dr. Jerry Rogers
Mississippi State University
P.O. Drawer EE
Mississippi State, MS 39762
(601) 325-3912 FAX (601) 325-2298
email: rogers@ee.msstate.edu

Mississippi State University is an Affirmative Action/Equal Opportunity Employer.

**COMPUTER SYSTEMS ANALYST**

M.S. in Computer & Information Science. Designs, implements and analyzes clients' business computer systems, including system capacities, networking, system layout, installation, modification, testing, and debugging, utilizing knowledge of management information system and computer science. Must be knowledgeable in Networks and Data Communication, Microcomputer Interfacing, and Structured System Design (1 yr. hands-on experience or 3 hrs. course work in each area). $26,716.00/yr., 40-hrs./wk. Contact Texas Employment Commission, Houston, Texas, or send resume to the Texas Employment Commission, TEC Building, Austin, Texas, 78778, J.O. #6122869. Ad Paid by An Equal Employment Opportunity Employer.

**SAN JOSE STATE UNIVERSITY**
**Department of Mathematics and Computer Science**

A tenure track faculty position in computer science is available for candidates holding a Ph.D. by August, 1991. The Ph.D. in computer science is preferred, but a Ph.D. in any of the mathematical sciences, together with substantial teaching/research experience in computer science, may also be acceptable. Rank and salary commensurate with experience.

The Department offers baccalaureate and masters degree programs in computer science, applied and pure mathematics, statistics, and mathematics education.

Applications will be reviewed beginning January 2, 1991 and will receive full consideration. Send vita, three letters of reference and transcripts to Dr. Veril Phillips, Chairman, Department of Mathematics and Computer Science, San Jose State University, San Jose, CA 95192-103. EEO/AA SCI 91-74.

**VALPARAISO UNIVERSITY**
**Computer Engineering Faculty**
**Tenure Track Position**

The Department of Electrical and Computer Engineering is seeking a qualified faculty member, holding a Ph.D. in Computer Engineering or a closely-related field, to teach in the area of Computer Engineering. This ABET accredited undergraduate program encompasses software and hardware with emphasis on architecture and embedded system applications. The successful candidate will join a team of dedicated teachers preparing students for entry level careers and graduate school admission in all areas of electrical and computer engineering. Send resume to Dr. Rodney J. Bohlmann, Valparaiso University, Valparaiso, IN 46383. Valparaiso University, affiliated with the Lutheran church, enrolls approximately 3500 students with 380 in engineering and is located 50 miles southeast of Chicago. AA/EOE. Applications from women and minorities are encouraged.

**TEXAS A&M UNIVERSITY**
**Department of Computer Science**

Applications are invited for faculty positions at the Assistant, Associate, or Full Professor level. Particular areas of interest include software engineering, databases, programming languages, computational sciences, and graphics, but outstanding candidates from all areas will be considered.

Texas A&M provides superior instructional and research facilities for its Computer Science faculty and is committed to a major expansion of its research and instructional program in Computer Science. The Department is a branch of Texas A&M's College of Engineering which is one of the nation's largest. Currently the Department has a roster of 28 full-time graduate faculty members with a number of new positions being added this year. In September of 1988 the Department initiated a program in Computer Science and Engineering to complement its degree offerings in Computer Science. In January of 1990 the Department moved into a new building with 50,000 square feet of space. The Department's equipment includes a 64 node NCUBE, a 2000 node MASPAR, Sequent Balance, numerous SPARC4, Silicon Graphics, Symbolics, NeXT, and real time system work stations as well as access to the University's Cray YMP2/116, IBM 3090/200E, Amdahl 5860, and more. The current annual external research funding in the Department is approximately $2.5 million.

The program seeks excellence in research. Applicants at the assistant professor level should show substantial promise for research and teaching. Applicants at the higher levels should show a strong record of research achievement. Ability in teaching graduates and undergraduates is essential. Applicants should have a doctoral degree or equivalent. Applicants should submit a resume and three references to Donald K. Friesen, Chairman, Faculty Search Committee, Computer Science Department, Texas A&M University, College Station, TX 77843-3112.

Texas A&M University is an equal opportunity/affirmative action employer.

## WEST VIRGINIA UNIVERSITY

The Department of Statistics and Computer Science (S/CS) is seeking applications for two tenure-track faculty positions as Assistant Professor of Computer Science beginning August 16, 1991. A Ph.D. in CS or equivalent is required. Applicants with research interest in AI, analysis of algorithms, databases, data communications, operating systems, computer graphics, programming languages, parallel computing and software engineering preferred. Duties include undergraduate- and graduate-level teaching; research and publication expected.

West Virginia University (WVU) is a comprehensive land grant state university of approximately 21,000 students. In addition to WVU there are substantial federal research facilities in the Morgantown area. Morgantown is a college town located in scenic northern West Virginia. Pittsburgh is about 1½ hours by car; Washington, D.C. is about a 3½ hour drive. There is air service to both Pittsburgh and Washington.

S/CS offers BS and MS degrees in both statistics and computer science and a Ph.D. degree in computer science. The Department has a strong working relationship with Carnegie Mellon University (CMU) and is actively involved in the Software Valley Initiative. The CMU-WVU Research Corporation was formed to enable CMU and WVU to jointly pursue funding for research centers to be located in the region.

Please send letter of application, resume, names & addresses of three references, and transcripts to Dr. Donald F. Butcher, Professor and Chairman, Statistics and Computer Science, 311 Knapp Hall, WVU, Morgantown, WV 26506. Initial consideration will be given to applications received prior to February 15, 1991. Applications will be accepted until suitable applicants are found. *WVU is An equal opportunity/affirmative action employer. Women and minorities are encouraged to apply.*

## OREGON STATE UNIVERSITY
**Department of Computer Science**

The Department of Computer Science invites applicants for tenure-track positions for Assistant, Associate, and Full Professorships. Specialization in computer graphics or software engineering is desirable, but all qualified applicants will be considered. Applicants should have completed or expect to complete all requirements for a Ph.D. in computer science or a closely related field and should have demonstrated research and teaching potential. Candidates for senior positions should have established research reputations. Review of applications will begin November 1, 1990, and will continue until the positions are filled. Please send vita, statement of research interests and plans, and three letters of reference to: Walter G. Rudd, Chairman, Department of Computer Science, Oregon State University, Corvallis, OR 97331.

Oregon State University is an equal opportunity affirmative action employer and complies with Section 504 of the Rehabilitation Act of 1973. OSU has a policy of being responsive to the needs of dual-career couples.

## THE UNIVERSITY OF MICHIGAN
**Department of Electrical Engineering and Computer Science**

The Department of Electrical Engineering and Computer Science at The University of Michigan invites applications for positions at all levels in its Computer Science and Engineering Division.

Our emphasis is on the areas of operating systems, distributed systems and networks, software engineering, programming languages, theoretical computer science, and database systems. Exceptional candidates in other areas of computer science and engineering will also be considered. All candidates who apply should have an interest in teaching and a strong research orientation.

Send your resume and the names of at least three references to Professor Bernard A. Galler, Chair of the Faculty Search Committee, CSE Division, EECS Department, The University of Michigan, Ann Arbor, MI 48109-2122.

The University of Michigan is an Equal Opportunity/Affirmative Action Employer.

## PORTLAND STATE UNIVERSITY
**Computer Science Department**
**Tektronix Professorship in Software Engineering**

The Tektronix Foundation has awarded Portland State University a $360,000 grant to upgrade its software engineering curriculum, establishing a new tenure-track faculty position in software engineering, with significant ancillary support. This new position can be at the junior or senior level. The new faculty member will join Dick Hamlet, Warren Harrison, Ralph London and Sergio Antoy on our faculty, and local software engineers such as Mayer Schwartz, to create a center for software engineering at PSU.

We invite applications and/or nominations for this position. Applicants must have an earned doctorate. Responsibilities include undergraduate and graduate teaching, development of sponsored research, and interaction with local industry. The position is available beginning Fall 1991.

Portland State University, one of the three major universities in the Oregon State System of Higher Education, is located in the heart of Portland, Oregon. The campus is downtown, near to parks, shopping, and the theater district. Portland is a beautiful city which offers a diversity of recreation within easy driving distance—unequaled fishing (salmon and steelhead within a mile of campus), skiing and mountain climbing, the scenic Oregon coast and unmatched state campgrounds, to name a few.

PSU's Computer Science Department is located in the Portland Center for Advanced Technology, which houses both the Electrical Engineering and Computer Science departments, plus CAD/CAM, VLSI design, computer vision and optical communications laboratories. The CS department operates a network of UNIX, AI, parallel processing and graphics systems and workstations.

Portland has a rapidly growing computer and electronics industry including Tektronix, Intel, Servio Logic, Sequent Computer Systems, Mentor Graphics, and Oregon Software, permitting close industry-university interaction. The excellent research facilities and faculty of the Oregon Graduate Institute are only a few miles away.

Send applications, including a resume and the addresses of three references, to:

Leonard Shapiro
Department of Computer Science
Portland State University
P.O. Box 751
Portland, OR 97207
Telephone: (503) 725-4036
len@cs.pdx.edu

Non-U.S. Residents must state their visa status. Portland State University is an equal opportunity/affirmative action employer. Minorities, women, and members of other protected groups are encouraged to apply. Deadline February 15, 1991 or until the position is filled.

## SOFTWARE ENGINEER

Point-of-sale software company in Cheney, WA seeks software engineer to direct (1) design, development, and implementation of communication software using SFX and 3780 Bisync protocol on UNIX workstations and PCs using C and (2) R&D on UNIX workstation, including software to interface to RF equipment and print UPC barcodes. Must have M.S. in Computer Science and 1 year experience in software engineering, including UNIX, UNIX internals, C, shell scripts, UNIX workstation hardware, COBOL, DOS, and Assembler. Must have proof of legal authority to work in the United States. Salary $34,800. 40 hrs/wk; 8 a.m. - 5 p.m. Submit resume by December 31, 1990 to: Employment Security Dept., ES Division, Job #235107-D, Olympia, WA 98504.

## FROSTBURG STATE UNIVERSITY
**Department of Computer Science**

Full-time, tenure track, Assistant/Associate Professor position available Fall 1991, SUBJECT TO FUNDING AND/OR APPROVAL TO HIRE. Teach undergraduate Computer Science and Computer Engineering or Computer Information Systems courses. REQUIRED: Master's degree in Computer Science, Information Science or related field. Ph.D. in Computer Science or related field preferred. Experience in Computer Engineering or Information Systems curricula development preferred. Successful candidate should have desire and ability to teach wide range of undergraduate Computer Science and Computer Engineering, or Computer Information Systems courses, as well as introductory level courses. Rank and salary dependent upon qualifications and experience. Position offers benefits package afforded University of Maryland System employees. Telephone inquiries to: Dr. Horton H. Tracy, Chair, 301-689-4361. Submit letter of interest, resume, transcripts and at least three letters of recommendation, not later than March 15, 1991, directly to: Mr. C. Douglas Schmidt, Director of Personnel Services, Frostburg State University, Frostburg, MD 21532. AA/EOE.

## NEW MEXICO STATE UNIVERSITY
**Computer Science Department**

We invite applications for tenure-track Assistant Professorships and visiting positions beginning Fall Semester 1991. Applications from all areas of Computer Science are welcome. Qualifications include a Ph.D. in Computer Science or closely related discipline and evidence of strength in teaching and research.

Degrees at BS, MS and Ph.D. levels are offered with 250 undergraduate and 50 graduate students currently enrolled. The department collaborates with the Computing Research Laboratory (CRL), an independent research center at NMSU. This interaction encourages research grant proposals, stimulates development of new research ideas and provides training for students in advanced computing. Departmental facilities include a network of some 30 SUN workstations, a 20 processor Sequent and access (through CRL and the Computer Center) to parallel machines from Thinking Machines, Floating Point Systems, Intel and IBM.

Las Cruces is situated in friendly southern New Mexico and has a metropolitan population of 90,000. The climate is totally agreeable; the campus is mountainously scenic. To apply, please send a resume and the names and addresses of three references to Dr. Juris Reinfelds, Head, Department of Computer Science, Box 30001, Dept. 3CU, New Mexico State University, Las Cruces, NM 88003. Ph. (505) 646-3724, e-mail juris@nmsu.edu. Inquiries regarding this re-advertisement may also be addressed to Dr. Roger Hartley, (505) 646-1218, e-mail rth@nmsu.edu. Applications will be welcomed until February 15, 1991.

## THE UNIVERSITY OF TENNESSEE
**Department of Computer Science**
**Knoxville, Tennessee 37996-1301**

The Department of Computer Science invites applications for tenure-track positions at the rank of Professor beginning Spring 1991. A strong research record in the areas of scientific computing, pattern and image analysis, compilers or operating systems is sought but all major fields in computer science may be considered. Experience directing doctoral students is especially important. Tenure-track positions for Associate and Assistant Professors are also open. Applicants for Associate Professor should have a strong research record, preferably in the above-named areas; experience directing doctoral students is desirable. Applicants for Assistant Professor should have a strong interest in research, preferably in the above-named areas. Applicants for all positions should have a doctoral degree in computer science or a related area.

Departmental SUN, IBM and DEC workstations abound for students and faculty and are fully networked. The department and the Mathematical Sciences Section of the Oak Ridge National Laboratory jointly operate the Advanced Computing Laboratory which includes fully networked Intel iPSC/860, 128 processors; iPSC/2, 64 processors; two Sequent Balances and a Sequent Symmetry; a Stardent Titan with four processors; Cogent; N-Cube; and various file servers. In addition, the department is part of the National Science Foundation Science and Technology center for Research in Parallel Computing. The University operates an IBM 3090 and a large VAX cluster.

Please respond to straight@utkvx.utk.edu. The mailing address is Department of Computer Science, 107 Ayres Hall, The University of Tennessee, Knoxville TN 37996-1301.

The University of Tennessee is an EEO/AA/TITLE IX/SECTION 504 employer.

## EASTERN OREGON STATE COLLEGE
**Computer Science Faculty**

A nine-month, tenure-track appointment in Computer Science at the assistant or associate professor level, beginning September 16, 1991. Rank and salary will be commensurate with academic preparation and experience. The successful candidate will be committed to teaching a wide range of undergraduate courses in computer science. An enthusiastic research involvement within the discipline is expected in an area appropriate to a college environment. A Ph.D. in computer science or closely related area is expected, although persons who are nearing completion of their Ph.D. will be considered. We prefer someone with a background and interest in operating systems, computing theory, software development, and database management. Ability to teach Pascal, the C programming language, and UNIX, are essential. Our facilities are excellent for our size. We are in a new building with well-designed laboratories emphasizing computer engineering, robotics, parallel processing, and computer graphics. EASTERN is located in the mountains of northeastern Oregon, with excellent opportunities for outdoor recreation. Send letter of application, cv, copies of publications, transcripts, and three letters of recommendation to: Chair, Computer Science Search Committee, Badgley Hall of Science, Eastern Oregon State College, La Grande OR 97850-2899. Application deadline: February 1, 1991, or until the position is filled. AA/EOE.

## ITHACA COLLEGE
**Department of Mathematics & Computer Science**

Ithaca College, Department of Mathematics & Computer Science invites applications for two tenure-eligible positions in computer science. Rank: at least Assistant Professor. Appointment effective: August 15, 1991. Duties include teaching a wide variety of courses in computer science. Ph.D. in computer science preferred; candidates with a Ph.D. in a closely related field and active ABDs with a completion by August 1992 will be considered. One position may require teaching courses in computer information science.

Submit letter of application, curriculum vita, and three letters of reference, at least one addressing teaching, to Dr. Stan Seltzer, Assistant Chair, Department of Mathematics & Computer Science, Ithaca College, Ithaca, New York 14850. Screening begins December 15, 1990. Ithaca College is an Affirmative Action/Equal Opportunity Employer.

## BALL STATE UNIVERSITY
**Muncie, Indiana**
**Announcement of Position Vacancy**
**Coordinator for Technology Assessment and Faculty Development**

Coordinates assessment of emerging computer-related technologies for the university's academic mission and goals. Serves as liaison between vendors, faculty, administrators, and Computing Services staff in the development of pilot projects and workshops involving advanced technologies; coordinates efforts within Computing Services and with the academic community to develop appropriate procedures for the assessment and absorption of new technologies in teaching, research, and administration. Minimum Qualifications: Masters degree or equivalent in computer science or related field, at least one year of development experience with micro and mainframe applications; at least three years experience with computer consulting in a college or university environment. Preferred Qualifications: Ph.D. or equivalent in computer science or related field, at least three years teaching experience in higher education. Send letter of application, vita, transcripts, and three letters of references to:

Dennis Kramer
University Computing Services
Ball State University
Muncie, IN 47306

Review of applications will begin immediately and continue until the position is filled.

Ball State University Practices Equal Opportunity in Education and Employment.

## BOISE STATE UNIVERSITY and HEWLETT-PACKARD

Boise State University, in cooperation with Hewlett-Packard, invites applicants for a tenure-track position in the Department of Mathematics for the fall of 1991. Applicants should have a Ph.D. in computer science or a related field with preference given to candidates with expertise in software engineering, artificial intelligence, or database systems. Industry experience is beneficial. Responsibilities will include teaching, research, and service split between the University and Hewlett-Packard's Boise site. Salary will be commensurate with rank and qualifications.

Boise State University is located in Boise, Idaho which is the state's largest city and is the site of many technology-based corporations. In the September issue of *Money* magazine, Boise was ranked 37th out of 300 of the best U.S. cities in which to live. The city is the political, economic, and cultural hub of Idaho. The person selected for this position can expect to find world-class outdoor recreational opportunities, an extraordinary quality of life, and a highly collegial faculty with balanced teaching and research interests.

A letter of application, a vita, three letters of reference, and graduate transcripts are required. Contact Dr. Phillip Eastman, Computer Science Search Committee Chair, College of Arts and Sciences, Boise State University, Boise, Idaho 83725. Screening will begin on January 15, 1991. BSU & HP are EEO/AA employers.

## UNIVERSITY OF MASSACHUSETTS AMHERST
### Faculty and Research Scientist Positions

The Department of Computer and Information Science invites applications for tenure-track faculty positions and nontenure-track research scientist positions at all levels in all areas of computer science. Applicants should have a Ph.D. in computer science or related area and should show evidence of exceptional research promise. Senior level candidates should have a record of distinguished research. Salary is commensurate with education and experience. Our Department has grown substantially over the past five years and currently has 29 full-time faculty, 12 research scientists, and 200 graduate students. Continued growth is expected over the next five years. We have ongoing research projects in robotics, vision, natural language processing, expert systems, distributed processing, database systems, information retrieval, real-time systems, software development, programming languages, computer networks, office automation, intelligent user interfaces, parallel computation, and computer architecture. The Department's NSF CER/CII award was recently renewed for a five-year program in robotics, computer vision and real-time computing. We also have a five-year DoD/URI Center of Excellence in Artificial Intelligence. To support our research, we have an extensive research computing facility, including over 200 Sun, VaxStation, DecStation and TI Explorer workstations, numerous servers, two Sequent Balance multiprocessors, a 4096-node Connection Machine, a variety of graphics devices, both Salisbury and Utah/MIT robotic hands, a Denning mobile robot and real-time testbed. Send vita, along with the names of four references to: Professor Allen Hanson, COINS Department, Lederle Graduate Research Center, University of Massachusetts, Amherst, MA 01003, by April 1, 1991. An Affirmative Action/Equal Opportunity Employer.

## ARIZONA STATE UNIVERSITY
### Computer Science
### Computer Engineering

The Department of Computer Science seeks outstanding faculty candidates for research and teaching in computer science and computer engineering. Applicants will be required to have completed a Ph.D. in computer science, computer engineering, or a closely related field by the date of appointment, and must show promise of excellence in teaching and research. All positions are tenure track, and candidates at all levels are invited to apply.

The department offers undergraduate and graduate programs through the Ph.D. The department is an active participant in an impressive alliance of high-tech companies (including Intel, Motorola, Bull-HN, Honeywell, DEC, Silicon Graphics, and AG Communications). The companies participating in this Engineering Excellence program are working with the University and the State to build an outstanding faculty and facilities for Computer Science and Computer Engineering at ASU. In 1992 the department will move into new facilities now under construction.

Faculty engineering workstations are networked locally to the Engineering Computing Service VAX, IBM, Convex, Harris, and Bull-HN mainframes, through the campus broadband network to the university Cray XMP/18se and IBM 3090/500E supercomputers, and externally via Internet. A major part of freshman/sophomore instruction is hosted on MacIntosh's, and PC/AT systems; all graphics laboratories are equipped with Silicon Graphics Iris 3130's; wide access is available to DEC-stations and Sun workstations. The research laboratories include Stardent, SGI, Xerox, and Symbolics workstations.

Please send a curriculum vitae, a selection of most important publications, and the names of three references to:

Dr. Ben M. Huey,
Faculty Search Committee
Department of Computer Science
Arizona State University
Tempe, Arizona 85287-5406
Internet: huey@asuvax.asu.edu

Deadline: January 31, 1991 and the last day of each month thereafter until filled. ASU is an EO/AA employer and encourages applications from women and minorities.

## DEPAUL UNIVERSITY CHICAGO, ILLINOIS
### Announcement of Positions in Computer Science

DePaul University invites applications for several tenure-track positions in computer science at all levels. The starting date is September, 1991. Any area of specialization will be considered; however persons in telecommunications and information systems will be given special consideration. Any applicant should hold a Ph.D. in computer science or a related field, or be a candidate for such a degree. Duties include a six-hour teaching load, advising, and research. Tenure details and salary are negotiable. Benefits include T1AA and standard health insurance. U.S. citizenship is not required.

The Department, which offers bachelor's, master's, and doctoral degrees, has over 500 undergraduate majors and over 800 graduate students. Facilities include a VAX 6000/410, a VAX 11/750, an IBM 4381, a Harris HCX-9, an AT&T 3B15, and a Harris 800. Each faculty office is provided with a high performance workstation connected to the Department's ethernet. In addition, the Department's Artificial Intelligence Laboratory is equipped with four Hewlett-Packard AI workstations, two Symbolics 3640s and a Symbolics 3670. The Department's Computer Vision and Graphics Laboratory is equipped with an AT&T 3B2-1000, eight AT&T 6386 WGS Model E workstations, 15 AT&T 630 multi-tasking graphics terminals, two frame grabbers, and a dedicated vision processor. There are also numerous PC laboratories. Faculty interests include telecommunications, information systems, artificial intelligence, computer vision, neural computing, natural languages, applied statistics, applied graph theory, computer graphics, computer security, compiler design, semantics of programming languages, and computer architecture.

Applications will be received until positions are filled. To apply, send a resume and at least three letters of reference to Helmut Epp, Chairman, Department of Computer Science and Information Systems, DePaul University, 243 S. Wabash, Chicago, IL 60604.

DePaul University is an equal opportunity employer.

## SENIOR SOFTWARE DESIGN ENGINEER

Senior Software Design Engineer needed to conduct high level design and research for advanced signalling technology including ISDN and CCS7. Design, implement and test software. Analyze data to determine feasibility of product proposal. Confer with research personnel to clarify or resolve problems and develop design. Plan and develop experimental test programs. Analyze data and reports to determine if design meets functional and performance specifications. Evaluate engineering test results for possible application to development of product. Requires a Bachelor's degree in Computer Science or its' equivalent and 4 years experience in job offered or 4 years directly related digital telecom signal design experience. Or will consider a Master's degree in Computer Science and 2 years of directly related telecommunications signalling design experience. 40 hour work week. $45,300 per year. Apply at the Texas Employment Commission, Richardson, Texas, or send resume to the Texas Employment Commission, TEC Building, Austin, Texas 78778, Job Order #5515178. Ad Paid By An Equal Employment Opportunity Employer.

## McGILL UNIVERSITY
### Computer Engineering

The establishment of a separate degree program in Computer Engineering has created a number of tenure-track faculty openings in the Department of Electrical Engineering at McGill University. Applications are invited from individuals who are dedicated to teaching at both the undergraduate and graduate level, and who have outstanding research potential and demonstrated research achievements. Practical experience in either Digital Systems or large Software Systems is essential.

Candidates must have an earned Ph.D. degree. Graduation from an accredited engineering school is desirable. Please send a resume and a list of 3 references to Professor Nicholas C. Rumin, Chairman, Department of Electrical Engineering, McGill University, 3480 University St., Montreal, QU, Canada, H3A 2A7.

In accordance with Canadian Immigration requirements, this advertisement is directed in the first instance to Canadian citizens and Permanent residents of Canada. Applications from others are welcomed, however, consideration of such candidates must be deferred until a Canadian search has been completed.

# Annual Index *Computer*

## Volume 23, 1990

This index covers all technical items — papers, correspondence, reviews, etc. — that appeared in this periodical during 1990, and items from previous years that were commented upon or corrected in 1990.

The *Author Index* contains the primary entry for each item, listed under the first author's name, and cross-references from all coauthors. The *Subject Index* contains several entries for each item under appropriate subject headings, and subject cross-references.

It is always necessary to refer to the primary entry in the *Author Index* for the exact title, coauthors, and comments/corrections.

† means to check the main author's entry for subsequent corrections and comments.

+ means to check the main author's entry for coauthors.

## AUTHOR INDEX

# SUBJECT INDEX

## USE THE READER SERVICE CARD TO OBTAIN INFORMATION ON:

- Membership application, student #203, others #202
- Perodicals subscription form for individuals #200
- Periodicals subscription form for organizations #199
- Publications catalog #201
- Compmail II electronic mail brochure #194
- Technical committee list/application #197
- Chapters lists, start-up procedures #193
- Student scholarship information #192
- Volunteer leaders/staff directory #196
- IEEE senior member grade application #204
  (requires ten years practice and significant performance in five of those ten)

To check membership status or report a change of address, call the IEEE toll-free number, 1-800-678-4333. Direct all other Computer Society related questions to the Publications Office.

## PURPOSE

The IEEE Computer Society advances the theory and practice of computer science and engineering, promotes the exchange of technical information among 100,000 members worldwide, and provides a wide range of services to members and nonmembers.

## MEMBERSHIP

Members receive the acclaimed monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

IEEE COMPUTER SOCIETY®

A member society of the Institute of Electrical and Electronics Engineers, Inc.

## PUBLICATIONS AND ACTIVITIES

**Computer.** An authoritative, easy-to-read magazine containing tutorial and in-depth articles on topics across the computer field, plus news, conferences, calendar, interviews, and product reviews.

**Periodicals.** The society publishes six magazines and five research transactions. Refer to membership application or request information as noted above.

**Conference Proceedings, Tutorial Texts, Standard Documents.** The Computer Society Press publishes more than 100 titles every year.

**Standards Working Groups.** Over 100 of these groups produce IEEE standards used throughout the industrial world.

**Technical Committees.** More than 30 TCs publish newsletters, provide interaction with peers in specialty areas, and directly influence standards, conferences, and education.

**Conferences/Education.** The society holds about 100 conferences each year and sponsors many educational activities, including computing science accreditation.

**Chapters.** Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

## OMBUDSMAN

Members experiencing problems — magazine delivery, membership status, or unresolved complaints — may write to the ombudsman at the Publications Office.

## EXECUTIVE COMMITTEE

| | |
|---|---|
| President: | Helen M. Wood* National Oceanic and Atmospheric Administration FB 4, Rm. 1069, Code E/SP Washington, DC 20233 (301) 763-1564 |
| President-Elect: | Duncan H. Lawrie* |
| Past President: | Kenneth R. Anderson* |
| VP, Conferences and Tutorials: | Laurel V. Kaleda (1st VP)* |
| VP, Standards: | Paul L. Borrill (2nd VP)* |
| VP, Area Activities: | Gerald L. Engel† |
| VP, Education: | Ronald G. Hoelzeman† |
| VP, Membership and Information | Barry W. Johnson† |
| VP, Press Activities: | James H. Aylor† |
| VP, Publications: | Sallie V. Sheppard* |
| VP, Technical Activities: | Mario R. Barbacci* |
| Secretary: | David Pessel* |
| Treasurer: | Joseph Boykin† |
| Division V Director | Edward A. Parrish, Jr.† |
| Division VIII Director: | J. T. Cain† |
| Executive Director: | T. Michael Elliott† |

*voting member of the Board of Governors
†nonvoting member of the Board of Governors

## BOARD OF GOVERNORS

**Term Expiring 1990:**
Vishwani Agrawal, Mario R. Barbacci, Ming T. (Mike) Liu, Yale N. Patt, Donald E. Thomas, Benjamin W. Wah, Ronald Waxman

**Term Expiring 1991:**
P. Bruce Berra, Michael Evangelist, Ted Lewis, Raymond E. Miller, Earl E. Swartzlander, Jr., Joseph E. Urban, Thomas W. Williams

**Term Expiring 1992:**
Alicja I. Ellis, Tadao Ichikawa, David Pessel, Sallie V. Sheppard, Bruce D. Shriver, Harold Stone, Wing N. Toy

**Next Board Meeting**
March 1, 1991, 8:30 a.m.
Cathedral Hill Hotel, San Francisco, CA

## SENIOR STAFF

| | |
|---|---|
| Executive Director: | T. Michael Elliott |
| Publisher: | H. True Seaborn |
| Director, Conferences and Tutorials: | Anne Marie Kelly |
| Director, Finance and Administration: | Tod S. Heisler |
| Director, Board and Administrative Services: | Violet S. Doan |

## COMPUTER SOCIETY OFFICES

**Headquarters Office**
1730 Massachusetts Ave. NW
Washington, DC 20036-1903
Phone: (202) 371-0101
Fax: (202) 728-9614

**Publications Office**
10662 Los Vaqueros Cir.
PD Box 3014
Los Alamitos, CA 90720-1264
Membership and General Information: (714) 821-8380
Publication Orders: (800) 272-6657
Fax: (714) 821-4010

**European Office**
13, Ave. de L'Aquilon
B-1200 Brussels, Belgium
Phone: 32 (2) 770-21-98
Fax: 32 (2) 770-85-05

**Asian Office**
Ooshima Building
2-19-1 Minami-Aoyama, Minato-ku
Tokyo 107, Japan
Phone: 81 (3) 408-3118
Fax: 81 (3) 408-3553

## IEEE OFFICERS

| | | | |
|---|---|---|---|
| President: | Carleton A. Bayless | VP, Educational Activities: | Richard S. Nichols |
| President-Elect: | Eric E. Sumner | VP, Professional Activities: | Michael J. Whitelaw |
| Executive Vice President: | Martha Sloan | VP, Publication Activities: | Ralph W. Wyndrum, Jr. |
| Secretary: | Fumio Harashima | VP, Regional Activities: | Robert T. H. Alden |
| Treasurer: | Wallace S. Read | VP, Technical Activities: | H. Troy Nagle, Jr. |

# The joy of C-scape

*Elegant graphics and text*

**The C-scape™ Interface Management System is a flexible library of C functions for data entry and validation, menus, text editing, context-sensitive help, and windowing. C-scape's powerful Look & Feel™ Screen Designer lets you create full-featured screens and automatically generates complete C source code.**

C-scape includes easily modifiable high-level functions as well as primitives to construct new functions. Its object-oriented design helps you build more functional, more flexible, more portable, and more unique applications—and you'll have more fun doing it.

**The industry standout.** Many thousands of software developers worldwide have turned to the pleasure of C-scape. The press agrees: "C-scape is by far the best. . . . A joy to use," wrote IEEE *Computer*. Major companies have selected C-scape as a standard for software development.

C-scape's open architecture lets you use it with data base, graphics, or other C and C++ libraries. C-scape runs in text or graphics mode, so you can display text and graphics simultaneously. To port from DOS or OS/2 to UNIX, AIX, QNX, or VMS, just recompile. C-scape also supports Phar Lap and Rational DOS extenders.

**Trial with a smile.** C-scape is powerful, flexible, portable, and easy to try. Test C-scape for 30 days. It offers a thorough manual and function reference, sample programs with source code, and an optional screen designer and source code generator. Oakland provides access to a 24-hour BBS, telephone services, and an international network of companies providing in-country support. No royalties, runtime licenses, runtime modules. After you register, you get complete library source code at no extra cost.

**Call 800-233-3733** (617-491-7311 in Massachusetts, 206-746-8767 in Washington; see below for International). After the joy of C-scape, programming will never be the same.

DOS, OS/2 (Borland and Microsoft support): with Look & Feel, $499; library only, $399; UNIX, etc. start at $999; prices include library source. Training in Cambridge and Seattle each month. Mastercard and Visa accepted.

**Graphics.** Run in color in text or graphics mode. Read images from PCX files.

**Object-oriented architecture.** Add custom features and create reusable code modules. C++ compatible.

**Mouse support.** Fully-integrated mouse support for menu selections, data entry fields, and to move and resize windows.

**Portability.** Hardware independent code. Supports DOS, OS/2, UNIX, AIX, VMS, others. Autodetects Hercules, CGA, EGA, VGA. Supports Phar Lap and Rational DOS extenders.

**Text editing.** Text editors with word wrap, block commands, and search and replace.

**Field flexibility.** Masked, protected, marked, required, no-echo, and named fields with complete data validation. Time, date, money, pop-up list, and many more higher-level functions; create your own.

**Windows.** Pop-up, tiled, bordered and exploding windows; size and numbers limited only by RAM.

**Menus.** Pop-up, pull-down, 123-style, or slug menus; create your own.

**Context-sensitive help.** Link help messages to individual screens or fields. Cross reference messages to create hypertext-like help.

**Code generation.** Build any type of screen or form with the Look & Feel™ Screen Designer, test it, then automatically convert it to C code.

**Screen flexibility.** Call screens from files at run time or link them in. Automatic vertical/horizontal scrolling.

**International support.** Offices in Berlin, Germany, with an international network of technical companies providing local training, support and consulting.

OAKLAND

**Oakland Group, Inc.** 675 Massachusetts Ave., Cambridge, MA 02139 USA. FAX: 617-868-4440. **Oakland Group, GmbH.** Alt Moabit 91-B, D-1000 Berlin 21, F.R.G. (030) 391 5045, FAX: (030) 393 4398. **Oakland International Technical Network** (training, support, consulting): **Australia** Noble Systems (02) 564-1200; **Benelux** TM Data (02159) 46814; **Denmark** Ravenholm (042) 887249; **Austria-Germany-Switzerland** ESM 07127/5244; **Norway** Ravenholm (02) 448855; **Sweden** Linsoft (013) 111588; **U.K.** Systemstar (0992) 500919. Photo by Jessica A. Boyatt; Kanji by Kaji Aso. Picture shows a C-scape program combining data entry with video images loaded from PCX files. C-scape and Look & Feel are trademarks of Oakland Group, Inc.; other trademarks belong to their respective companies. Copyright © 1990, by Oakland Group, Inc. Features, prices, and terms subject to change.

*For the latest UNIX research & development topics . . .*

# Winter 1991 USENIX Technical Conference

**Grand Kempinski Hotel**

**Dallas, Texas**

**January 21-25, 1991**

*The Professional and Technical UNIX Association*

USENIX

For complete conference details, call: (714) 588-8649 or write:
USENIX Conference Office
22672 Lambert St., Suite 613
El Toro, CA 92630

## TUTORIALS

Two days of in-depth tutorials will be offered by internationally renowned UNIX developers and educators. Software professionals and technical managers who want to broaden their expertise in advanced computing systems should register early!

- System Security
- UNIX Systems Internals
- Introductory and Advanced Courses on C++
- Gnu C Compiler Internals
- Graphics and X Windows
- Parallel Programming
- UNIX Technologies in Japan
- Systems Administration & Perl
- Mach
- TCP/IP
- Network Programming & Security

## TECHNICAL SESSIONS

Presentations by leading experts on the evolution and development of the UNIX operating system will feature:

- Operating Systems of the Future
- Communications and Networking
- Applications
- User Interfaces
- Programming Environments and Languages
- Testing and Debugging

Special interest sessions offer the attendees an opportunity to exchange information and ideas on such topics as systems administration, security and networking. Facilities featuring Internet and dialout access for attendees is provided.

## THE SPONSOR

The USENIX Association is a not-for-profit professional organization dedicated to fostering the development of research and technological information pertaining to UNIX and advanced computing systems.

*The 1991 UniForum Trade Show will be held at the Dallas Infomart on the same dates as the USENIX Conference.*